Sonderheft Nr. 216
Preis DM 19.50–
öS 150.–, sfr 19.50–

Elektronik

Schaltungs-praxis

Mikro-prozessor-schaltungen

Steuer- und Regel-schaltungen

Meß-, Prüf- und Sensor-schaltungen

Wandler-schaltungen

Mikro-prozessor-schaltungen

Digital- und Logik-schaltungen

Analog-, Operations-verstärker, Strom-versorgung

Schaltungsideen für Elektroniker

Informationen für Produkt-Entwickler

Die ELEKTRONIK liefert Ihnen Kompaktinformationen zur Produktentwicklung und zum industriellen Einsatz, die neben der Technik auch die Marktsituation beschreiben.

In Fachaufsätzen finden Sie Anregungen für die Entwicklungsarbeit z. B. auf den Gebieten Bauelemente, Meß- und Prüftechnik, Computer und Software, CAE, Automatisierung, Schaltungspraxis.

Die ELEKTRONIK informiert unabhängig und objektiv. ELEKTRONIK-Redakteure sind praxiserfahrene Ingenieure, Physiker und Informatiker. In der ELEKTRONIK berichten Kollegen aus der Industrie, aus Entwicklungslabors, Hochschulen, Instituten und Vertriebsabteilungen.

ELEKTRONIK-Beiträge werden Sie direkt nutzen. Die ELEKTRONIK erscheint alle 14 Tage, das Jahresabonnement kostet DM 126,–. Ein kostenloses Probeheft schicken wir Ihnen gerne im Rahmen unseres Kennenlern-Angebotes. Die vorbereitete Karte finden Sie an der Umschlagklappe dieses Heftes.

Elektronik

Fachzeitschrift für Entwickler und industrielle Anwender.

Vorwort

Schaltungsideen als Denkanstöße

Dem Wunsch vieler ELEKTRONIK-Leser, mehr Schaltungspraxis-Beiträge zu veröffentlichen, ist die ELEKTRONIK-Redaktion in den letzten Jahren in erheblich gesteigertem Maße nachgekommen. Dies war allerdings nur möglich dank der aktiven Mitarbeit vieler Leser, die ihre Schaltungsideen – anstatt sie in der Labortisch-Schublade „schlummern" zu lassen – schriftlich niederlegten und sie der Redaktion zur Veröffentlichung anboten. Daß man auf diese Weise sein Einkommen ein bißchen aufbessern kann, versteht sich von selbst.

Die große Zahl der erschienenen Beiträge versetzte die Redaktion in die Lage, dieses Mal mehrere Sonderhefte zu ganz spezifischen Themengebieten zusammenzufassen. So entstand eine Folge von insgesamt sieben Sonderheften, wodurch der Kauf-Interessent die Möglichkeit hat, stets dasjenige Sonderheft auszuwählen, das er für seine Arbeit benötigt. Die sieben Hefte sind nach folgenden Themengebieten gegliedert:

1. Mikroprozessorschaltungen inkl. Software
2. Steuer- und Regelschaltungen
3. Meß-, Prüf- und Sensorschaltungen
4. Interface- und Wandlerschaltungen
5. Digital- und Logikschaltungen
6. Oszillatoren, Generatoren, Rechen- und Filterschaltungen
7. Analog-, Operationsverstärker- und Stromversorgungsschaltungen

Sie werden in der angegebenen Reihenfolge erscheinen.

Die Beiträge des vorliegenden Sonderheftes „Mikroprozessorschaltungen" stammen vollständig aus den Rubriken „Schaltungspraxis, µC-Praxis, Applikation und Software" der Fachzeitschrift ELEKTRONIK. Die Auswahl wurde so getroffen, daß der praktisch tätige Ingenieur oder Elektronik-Techniker, dessen tägliches Brot die Konzeption neuer Schaltungen ist, für seine Arbeit eine wertvolle Unterstützung findet und Entwicklungszeit und Kosten einspart. Es sind einfache Schaltungen von großer Anwendungsbreite enthalten, die zum Teil so allgemeiner Art sind, daß sich für viele Probleme zumindest Denkanstöße ergeben. Einige Schaltungen bieten darüber hinaus Lösungen zu speziellen Problemen an. – Allen ist gemeinsam, daß sie aus der Praxis der industriellen Elektronik stammen.

Wir sind sicher, daß Sie in Ihrem beruflichen Alltag größtmöglichen Nutzen aus diesem Heft ziehen. Können Sie von den beschriebenen Schaltungsideen auch nur eine brauchen, so hat sich der Kauf des Heftes für Sie vielleicht schon gelohnt. – Ergibt sich für Sie nicht auch der Anreiz, selbst einmal einen Beitrag für die Rubriken „Schaltungspraxis" und „µC-Praxis" zu schreiben?

Ihre
ELEKTRONIK-Redaktion

Inhalt

Schnelle Arithmetik mit 80186/8087-Kombination 5
„Breakpoint"-Schaltung vereinfacht 8
Aus 8085 wird Z80 10
Bildschirmsteuerung mit EPROMs 11
Analogsignalverarbeitung mit μC 13
Time-Sharing-Betrieb für Mikroprozessoren 14
Meßwerte direkt in den Speicher gebracht 17
Eine CPU-Platine mit dem iAPX 286
muß nicht teuer sein 20
Serielle Ausgabe-Erweiterung für
Einchip-Mikrocomputer 24
Adreß-Tracer 26
Mikrocontroller besitzt BASIC-Interpreter 28
Dynamische RAMs ohne Wartezyklen 34
Einchip-Computer steuert Frequenzzähler 37
Resetlogik, Spannungsüberwachung und Watchdog . 39
6809-Systeme preiswert getestet 40
Komplexe Multiplikation in kurzer Zeit 43
Verkürzter WAIT-Zyklus beschleunigt
68000-Systeme 44
Dynamisches RAM für 68000-Systeme 46
Kassettenrecorder zeichnet Signale sehr
niedriger Frequenz auf 48
Verbesserte Einzelschrittschaltung für Z80 52
Einfaches Interface zwischen dynamischen
RAMs und Z80-Systemen 55
Einfache Interrupt-Schaltung für Z80-Systeme 58
Schneller Multiplizierer für Mikrocomputer 59
CPU schneller als der Bus 62
Beliebiger Speicherbereich für AIM-65-BASIC 63
Watchdog-, Reset- und Zeitbasisschaltung 64
„Watch-dog"-Schaltungen erkennen
μP-Systemstörungen 66
Mikrocomputer fragt Multifunktionszähler ab 69
Adressenextensor für 8-Bit-μP 71
Adressengenerator für den S-100-Bus 73
Memory-Mapping für Mikroprozessoren 74
Erzeugung netzspannungsgesteuerter
Programmunterbrechungen 75
Automatisches Umschalten eines Druckers
zwischen mehreren Rechnern 76
Der Schutz von Mikrocomputern
gegen elektrische Störungen 77
Wurzelziehen durch sukzessive Approximation 79
Zahlenumwandlung in PASCAL 80
Grafikprozessor in MPL programmiert 80
BASIC-Programm simuliert PLL-Schaltung 83
Universeller Quadratwurzel-Algorithmus 86
Schnelle Fouriertransformation in BASIC 88
Beschleunigung der Multiplikation ohne
zusätzliche Hardware 90
8080 – 8085 – Z80: Nicht ganz kompatibel 93
Quadratwurzel mit dem μP Z80 94
BASIC-Programm berechnet Fehler eines
D/A-Umsetzers 96
Programm zur Gray-Code-Decodierung 97

1986
Franzis-Verlag GmbH, Karlstraße 37–41, 8000 München 2.
Bearbeitet von der Redaktion der Zeitschrift ELEKTRONIK. Für den Text verantwortlich: Günther Klasche.

© Sämtliche Rechte – besonders das Übersetzungsrecht – an Text und Bildern vorbehalten. Fotomechanische Vervielfältigung nur mit Genehmigung des Verlages.
Jeder Nachdruck, auch auszugsweise, und jede Wiedergabe der Abbildungen, auch in verändertem Zustand, sind verboten.

Druck: Franzis-Druck GmbH, München. Printed in Germany. Imprimé en Allemagne. ZV-Artikel-Nr. 21602 · F/MK/886/1210/6' · ISSN 0170-0898

Schnelle Arithmetik mit 80186/8087-Kombination

Um auch komplexe arithmetische Funktionen auf Chipebene ausführen zu können, entwickelte Intel vor einiger Zeit passend zur CPU 8086 den Fließkomma-Coprozessor 8087. Eine Kombination des 8087 mit dem höherintegrierten Mikroprozessor 80186 war bisher aufgrund des unterschiedlichen Busprotokolls beider Bausteine nur mit einem aufwendigen Hardware-Interface möglich. Seitdem der Interface-Baustein 82188 zur Verfügung steht, ist eine Kopplung 80186/8087 kein Problem mehr.

Der 8087 ist ein numerischer Coprozessor zur iAPX-86-Familie von Intel und erlaubt die Berechnung arithmetischer und komplexer Funktionen in verschiedenen Datenformaten nach dem IEEE-Floating-Point-Standard P 754 (*Bild 1*). Er enthält intern acht Register mit jeweils 80 Bit für Real-Zahlen (64-Bit-Mantisse, 15-Bit-Exponent mit Vorzeichen, 1-Bit-Exponent) sowie mehrere Status- sowie Steuerregister und unterstützt folgende Datentypen:

- Word Integer (16 Bit)
- Short Integer (32 Bit)
- Long Integer (64 Bit)
- Packed Dezimal (60 Bit)
- Short Real (32 Bit)
- Long Real (64 Bit)
- Temporary Real (80 Bit)

Intern werden sämtliche Operationen im sogenannten „Temporary-Real-Format" ausgeführt, so daß praktisch keine Rundungsfehler entstehen. Um eine von der Host-CPU unabhängige Bearbeitung der arithmetischen Funktionen zu gewährleisten, verfolgt der 8087 die Status- (S0, S1, S2) und Queue-Status-Leitungen (QS0, QS1) parallel zur CPU. Erkennt der 8087 anhand dieser Signale einen für ihn bestimmten Befehl (Prefix ESC im Assembler-Code), führt er diesen parallel zur Haupt-CPU in einem oder mehreren seiner internen Register aus.

Solange der 8087 mit der Ausführung von Rechenoperationen beschäftigt ist, hält er seine Busy-Leitung aktiv, was von der CPU 8086 über den Test-Eingang im Zusammenhang mit einem Wait-Befehl abgefragt werden kann. Für die Übertragung von Daten zwischen CPU und 8087 wird über das Request-/Grant-Handshake-Verfahren (RQ/GRT) der lokale Speicher als Mailbox entweder für die Haupt-CPU oder für den 8087 freigegeben.

Eine Kombination des 8087 mit dem Mikroprozessor 80186, der zwei DMA-Kanäle, drei 16-Bit-Timer, Taktgenerator und Chip-Select-Logic auf dem Chip enthält,

Bild 1.
Datentypen des 8087

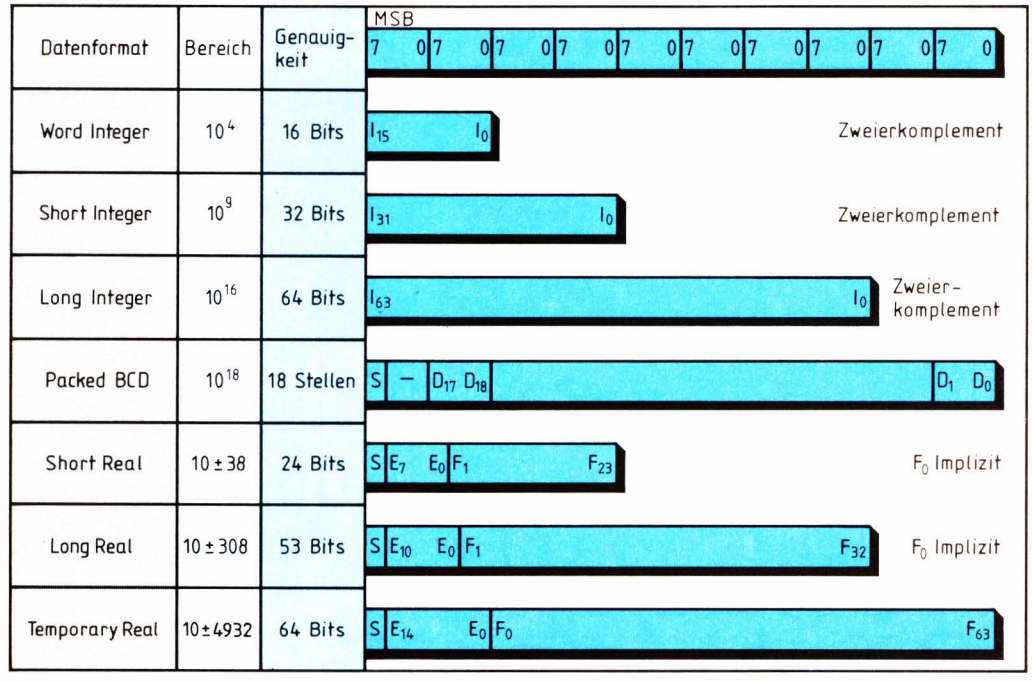
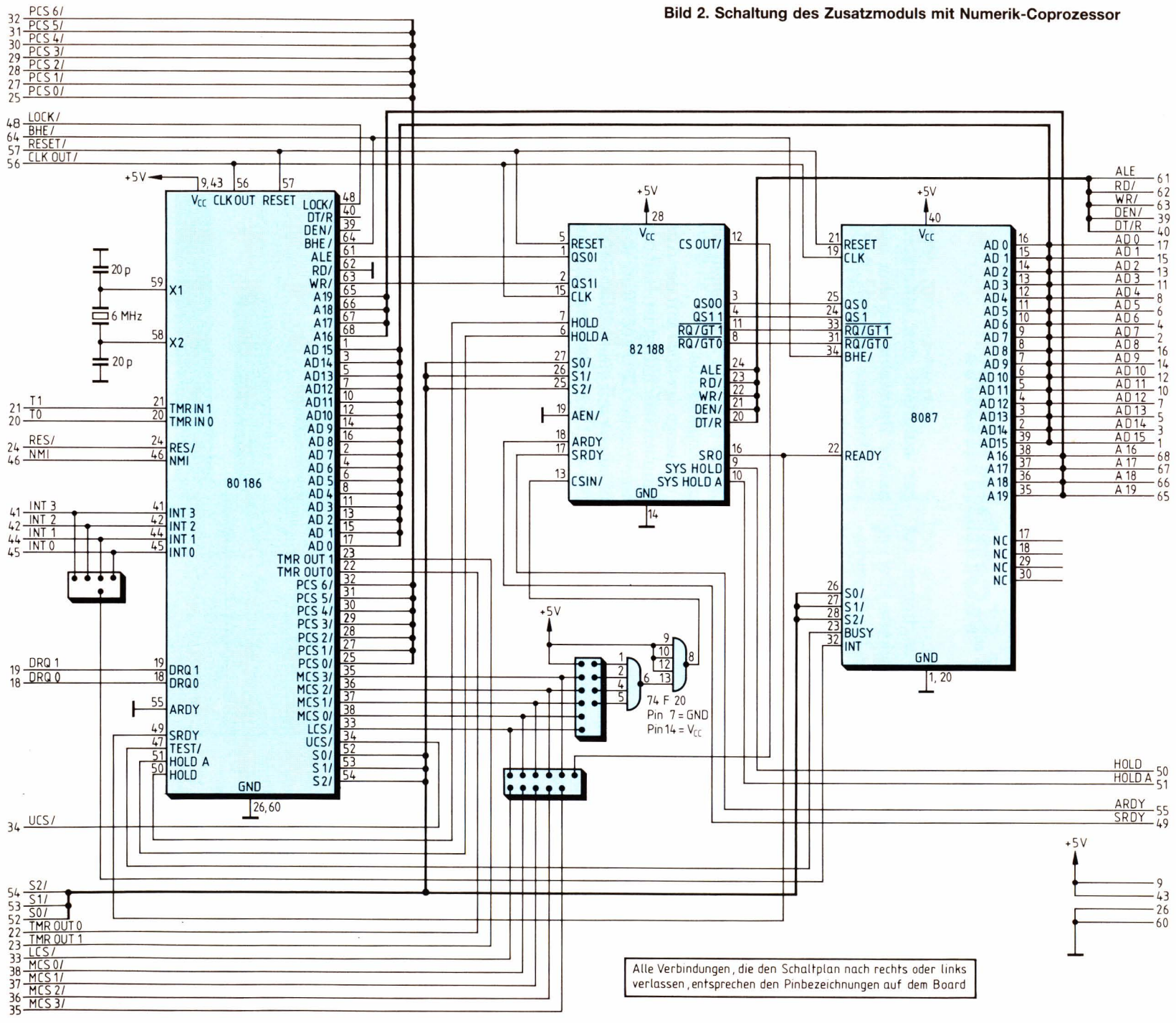


Bild 2. Schaltung des Zusatzmoduls mit Numerik-Coprozessor



war bisher nur mit einer aus etwa 20 Bausteinen aufgebauten TTL-Logik möglich.

Die Schwierigkeit besteht nämlich darin, folgende Differenzen der beiden Bausteine aneinander anzupassen:

- Unterschiedliches HOLD/HOLDA- (80186) und RQ/GRT-Protokoll (8087).
- Takttastverhältnis 50 % (80186) gegenüber 33 % (8087).
- Queue-Status-Signale des 80186 erscheinen ½ Takt früher als beim 8087.
- Der 80186 fügt nach dem RESET drei Wait-States ein, bis die internen Register programmiert sind.
- Der 80186 kann intern auf bis zu drei Wait-States programmiert werden.

Der neue Interface-Baustein 82188 beseitigt all diese Differenzen (ausgenommen das unterschiedliche Tastverhältnis), so daß eine problemlose Kombination 80186/8087 möglich wird. Lediglich beim letzten Punkt ist noch etwas Vorsicht geboten; der Speicherbereich, auf den auch der 8087 zugreift, darf nur extern erzeugte Wait-States generieren, da der 8087 intern keine Wartezeiten erzeugen kann.

Hardware

Bild 2 zeigt die komplette Schaltung 80816/82188/8087. Wie man sieht, ist keine zusätzliche Logik erforderlich. Das NAND-Gatter dient lediglich zur Verknüpfung mehrerer CS-Signale des 80186, um evtl. einen über den programmierbaren Bereich hinausgehenden Speicherbereich zu adressieren. Falls nur eine CS-Leitung des 80186 (z. B. LCS für den Bereich 0...128 KByte) verwendet wird, kann auch dieses Gatter entfallen. Aufgrund des verschieden spezifizierten Tastverhältnisses (50 % beim 80186, 33 % beim 8087), muß der 8087 in dieser Kombination immer für eine höhere Taktfrequenz spezifiziert sein als der 80186. Ein 80186 in 6-MHz-Ausführung benötigt also einen 8-MHz-Baustein 8087, eine 8-MHz-CPU 80186 eine 10-MHz-FPU 8087. Da der 8087 die Queue-Status-Information des 80816 benötigt, muß der 80816 im Queue-Status-Mode (Anschluß 62, RD = Low) betrieben werden.

Bild 3 zeigt ein fertiges „Piggy-Pack"-Modul, das in fast jede existierende 80186-Schaltung statt der CPU 80186 eingesteckt werden kann. Aus Gründen der kapazitiven Belastung wurde der Quarz direkt auf dem Modul untergebracht.

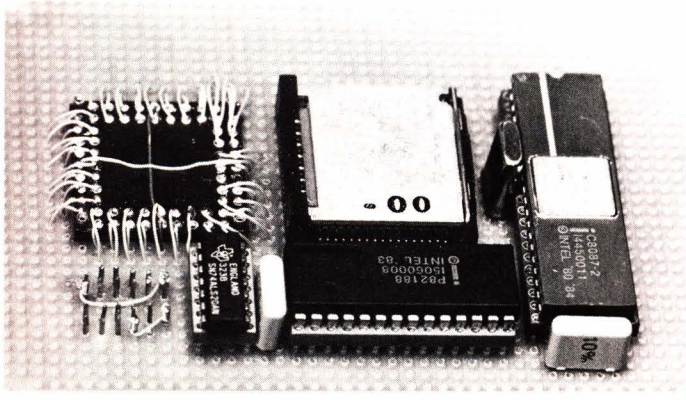


Bild 3. Dieses Zusatzmodul kann anstelle des 80186 auf eine Rechnerplatine gesteckt werden, um die Verarbeitungsleistung für numerische Probleme zu erhöhen

Funktion

Bild 4 zeigt ein kleines Assemblertestprogramm zur Berechnung einer Quadratwurzel aus einer Integer-Zahl.

```
START:  FNINIT                      ; INIT 8087
        MOV     BX,OFFSET INTEGER
        FILD    WORD PTR [BX]       ; LOAD INTEGER INTO STACKREGISTER 0
        FST     ST(1)               ; MOVE INTEGER INTO STACKREGISTER 1
        FSQRT                       ; SQUARE ROOT
        MOV     BX,OFFSET SQRT
        FIST    WORD PTR [BX]       ; STORE INTEGER
        FWAIT                       ; CPU WAIT
        END
```

Bild 4. Assembler-Testprogramm

Die Adresse der Variablen Integer wird im Register BX abgelegt und dem 8087 über den Word-Pointer (BX) übergeben. Der 8087 lädt den Inhalt der über den Pointer adressierten Speicherzelle in sein ST0-Register, schiebt es in ST1, berechnet aus ST0 die Wurzel und übergibt das Ergebnis an den Speicher zurück. Dieses kleine Testprogramm reicht völlig aus, um die richtige Funktion der Hardware zu zeigen.

Zur Inbetriebnahme der Hardware bieten sich grundsätzlich zwei Möglichkeiten an:

- Testen mit einem Monitorprogramm, z. B. dem Monitor SDM86, der für ein 80186/8087-System konfigurierbar ist und sämtliche Operationen des 8087 überprüfen kann.

- Debugging mit dem In-Circuit-Emulator I2ICE, der mit Probes für den 8086/8088, 80186/80188 und 80286 erhältlich ist.

Aufgrund seiner Merkmale, wie symbolisches Hochsprachen-Debugging, Trace-Speicher und integriertem Logikanalysator, stellt er eindeutig die komfortablere,

allerdings auch teurere Lösung dar. *Bild* 5 zeigt ein I2ICE-Testprogramm, das speziell zum Austesten des Piggy-Pack-Moduls erstellt wurde.

Geschwindigkeitsvergleich

Um die mögliche Steigerung der Rechengeschwindigkeit zu demonstrieren, wurde mit einem Personal Computer Alphatronic P60 (6-MHz-80186) folgendes Turbo-Pascal-Benchmark durchgeführt:

```
program wurzel;
var
  e: real
  l: integer;
begin
  for 1:=1 to 2000 do
    begin
      e:=sqrt(1)*sqrt(1);
    end;
  writeln (e);
end.
```

```
RESET UNIT
MAP OFFC00H LENGTH   1K HS
MAP 0H      LENGTH   4K USER
CSCTRL(2)=00F8H
LOAD :F8:TEST87

DEFINE LITERALLY L='LENGTH'
DEFINE LITERALLY ST = 'OFFFFH:0'
ASM H L 4
BYTE 401 L 10=00

QSTAT=TRUE
CPMODE=2
GET87(700H)

DEFINE PROC S=D0
ISTEP
ASM H
REGS
PINS
ST0
ST1
END
```

Bild 5. In-Circuit-Emulator-Testprogramm

Die Ausführungszeit beträgt hierbei nur 1(!) s mit 8087 im Gegensatz zu 64 s ohne den Fließkomma-Coprozessor 8087.

Zusammenfassend kann gesagt werden, daß mit dem Erscheinen des Interfacechips 82188 eine Nachrüstung bestehender 80186-Hardware mit dem Floating-Point-Coprozessor 8087 kein Problem mehr ist. Für neue Entwicklungen läßt sich mit minimalem Aufwand eine Kombination von hoher Integration an Peripherie (80186) mit schneller Arithmetik (8087) erreichen.

Thomas Criegee

„Breakpoint"-Schaltung vereinfacht

Viele Mikrocomputersysteme bieten zwar die Möglichkeit, ein selbsterstelltes Programm im Einzelschritt zu durchfahren. Bei längeren Programmabschnitten ist dies jedoch ziemlich zeitaufwendig. Auch das Verfahren mit sogenannten Software-Breakpoints ist reichlich umständlich. Der Break-Befehl muß jedesmal wieder durch den ursprünglichen Befehl ersetzt werden.

Mit der im *Bild* gezeigten Schaltung können Programme sowohl im RAM- als auch im ROM-Bereich unterbrochen werden. Durch Löschen des „Breakpoint" wird das Programm an gleicher Stelle fortgesetzt.

Die Schaltung

Die Funktionsweise ist einfach: Eine gespeicherte Adresse wird ständig mit der Adresse auf dem Adreßbus verglichen. Sind beide identisch, wird ein Interrupt ausgelöst. Die Interruptroutine muß die Register sichern und anschließend auf einen neuen Startbefehl warten.

Jedes 8-Bit-System mit zwei Ports kann mit dieser Schaltung arbeiten. Von den 16 Latches (2 × 74LS374) liegen je acht eingangsseitig an einem Port. Vom zweiten Port werden noch drei Leitungen benötigt.

Mit PB0 und PB1 werden das obere und das untere Byte der Adresse, an der unterbrochen werden soll, in die Latches geschrieben. Mit der ansteigenden Flanke der PB-0- und PB-1-Signale an den Takteingängen der Latches werden die Informationen nacheinander übernommen. Damit steht die Vergleichsadresse auch an den 16 Exklusiv-ODER-Gattern an, von denen jeweils der zweite Eingang mit dem Adreßbus des Systems verbunden ist. Entspricht nun die Adresse des Adreßbusses der gespeicherten Adresse, liegt an allen Ausgängen der Exklusiv-ODER-Gatter Low-Potential. Damit wechseln die Ausgänge der sechs NOR-Gatter auf High. Über das NAND-Gatter mit acht Eingängen werden die sechs Signale verknüpft. Am Ausgang des NAND-Gatters liegt Low-Potential, wenn die zwei „freien" Eingänge auf High liegen. Sie arbeiten als Enable-Eingänge. Da sich die Eingangs-Latches immer in einem bestimmten Zustand zwischen 0000 und FFFF befinden, kann ein Interrupt durch Low-Pegel an PB2 unterbunden werden. Mit diesem Disable-Signal wird auch der „Breakpoint" gelöscht.

Die Software

Zum Setzen des „Breakpoint" ist natürlich auch etwas Software nötig. Nachfolgend sind die Schritte angegeben, die notwendig sind:

- PA 0...7 und PB 0...2 als Ausgang definieren.
- An PA das untere Byte der Break-Adresse anlegen und durch High-Pegel an PB0 speichern.
- An PA das obere Byte der Break-Adresse anlegen und durch High-Pegel an PB1 speichern.

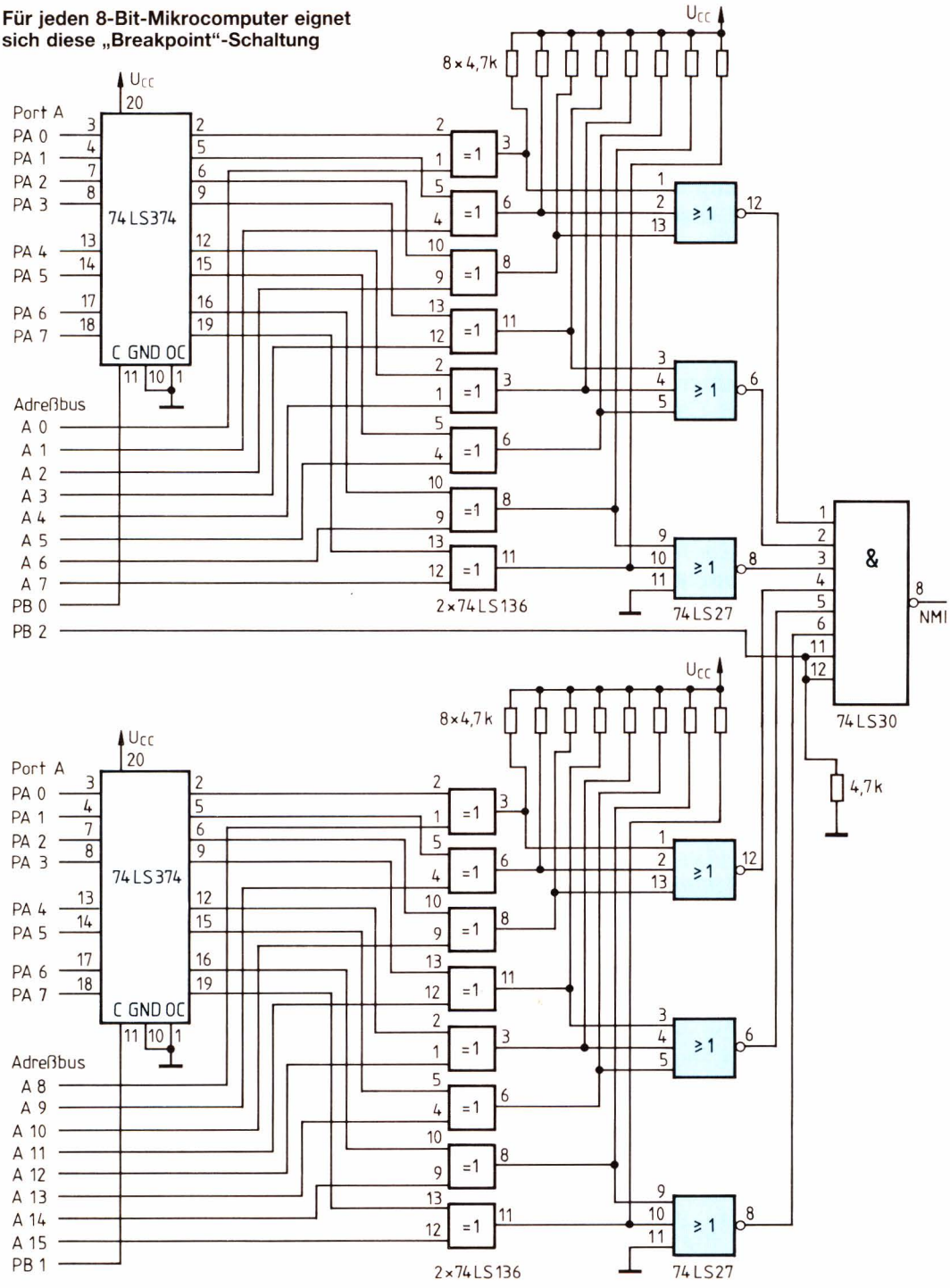
– PB 2 = High aktiviert das NAND-Gatter 74LS30.
– PB 0 und PB 1 auf Low legen (als Vorbereitung für den nächsten Interrupt).

„Breakpoint" löschen

Zum Löschen des „Breakpoint" ist nur ein einziger Schritt nötig. An PB 2 muß Low liegen. Damit wird ein Interrupt verhindert. Außer den genannten Schritten ist ein Interruptprogramm zu schreiben, das alle Register sichert und auf eine neue Eingabe wartet, die das Programm erneut startet. Der Interruptvektor wird beim Setzen des „Breakpoint" auf diese Routine gerichtet und beim Löschen wieder auf den ursprünglichen Wert gesetzt.

Heinz Wieking

Für jeden 8-Bit-Mikrocomputer eignet sich diese „Breakpoint"-Schaltung



Aus 8085 wird Z80

Da der Befehlssatz des Mikroprozessors 8085 nur eine Teilmenge (ausgenommen SIM und RIM) des viel umfangreicheren Z80-Befehlssatzes bildet, können zwar 8085-Programme auf Z80-Systemen ablaufen, die Übernahme von Z80-Software auf 8085-Systeme ist jedoch nicht bzw. nicht direkt möglich. Im folgenden wird ein einfacher Weg aufgezeigt, der die Verarbeitung von Z80-Programmen in 8085-Mikrocomputern ermöglicht.

Der Schlüssel zur Lösung dieser Aufgabe ist der CMOS-Mikroprozessor NSC800 von National Semiconductor. Dieser Prozessor weist ein (fast) Z80-gleiches Innenleben auf und ist nach außen hin dem 8085 sehr ähnlich. Die internen Unterschiede gegenüber dem Z80 bestehen in dem auf 8 Bit erweiterten Refresh-Register des NSC800 und in der fixen internen Verwendung der I/O-Adresse BB_H als Interrupt Control Register (ICR). (Durch Setzen entsprechender ICR-Bits wird die Annahme der vier maskierbaren Interrupts $\overline{RSTA}$, $\overline{RSTB}$, $\overline{RSTC}$ und $\overline{INTR}$ ermöglicht.) Nach außen hin differieren 8085 und NSC800 in folgenden Punkten:

a) Anschlußschema
b) logische Aktivierungspegel diverser Leitungen
c) serielle Ein-/Ausgänge.

Wie aus den Anschlußbildern in den entsprechenden Datenblättern hervorgeht, weisen NSC800 und 8085 achtunddreißig funktionell gleiche Anschlüsse auf, nur sind sie unterschiedlich plaziert. Neben Anordnungsunterschieden bestehen aber auch noch Differenzen in den logischen Aktivierungspegeln diverser Leitungen, einer Interruptadresse und der Funktion des Interrupteingangs niedrigster Priorität (*Tabelle*).

Außer den achtunddreißig funktionsgleichen Anschlüssen von 8085 und NSC800 gibt es lediglich zwei, die funktionell völlig verschieden sind. Es sind dies $\overline{PS}$ (power save input) und $\overline{RFSH}$ (refresh output) beim NSC800 und SID (serial input data) und SOD

Tabelle der funktionellen Unterschiede zwischen 8085 und NSC800

8085		NSC800		
An-schluß	Restart-Adresse	An-schluß	Restart-Adresse	Funktion
TRAP	24H	$\overline{NMI}$	66H	Interrupt höchster Priorität. Nicht maskierbar.
RST 7,5	3CH	$\overline{RSTA}$	3CH	Interr. zweithöchst. Priorität. Maskierbar.
RST 6,5	34H	$\overline{RSTB}$	34H	Interr. dritthöchster Priorität. Maskierbar.
RST 5,5	2CH	$\overline{RSTC}$	2CH	Interr. vierthöchster Priorität. Maskierbar.
INTR		$\overline{INT}$		Interr. niedrigster Priorität. NSC800: drei Betriebsarten wie Z80.
HOLD		$\overline{BREQ}$		Busfreigabeaufforderung
HLDA		$\overline{BACK}$		Busfreigabesignal

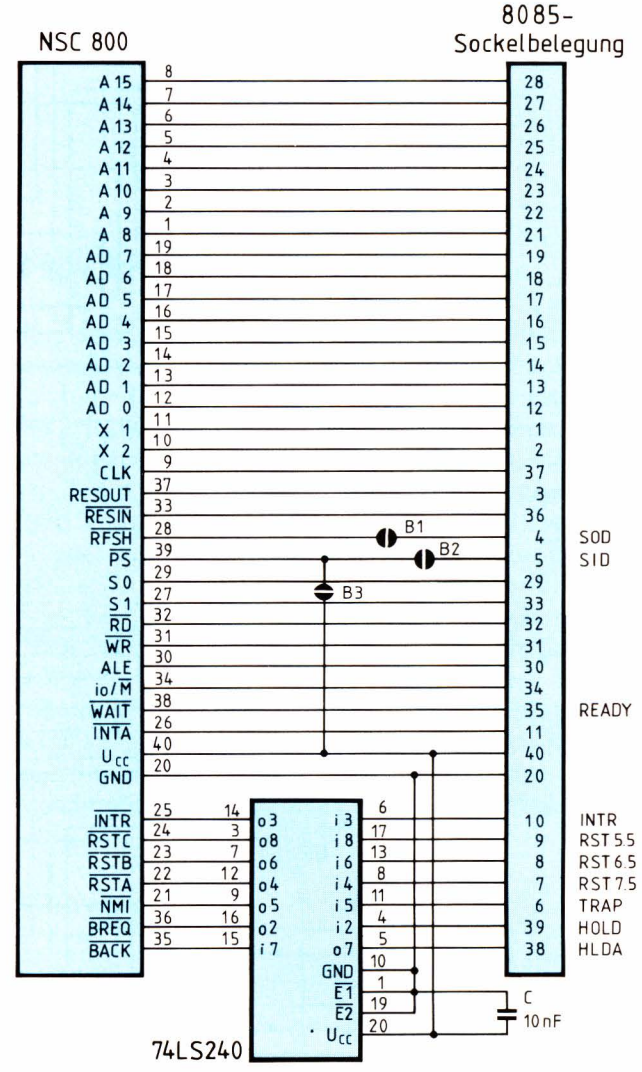


Bild 1. Schaltung der Adapterplatine mit NSC800 und Inverter. B1...B3 sind Brücken (B3 ist im Regelfall geschlossen, Power-Save-Eingang auf U_{CC}). Die Systemtaktfrequenz ist dem verwendeten NSC800 anzupassen

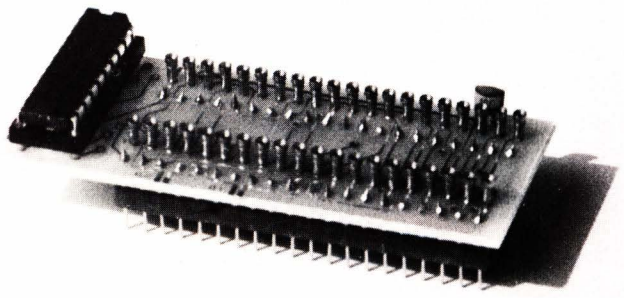


Bild 2. Die Adapterplatine ohne NSC800 aber mit eingesetztem Inverterbaustein; auf der Unterseite der 40polige Stecker als Verbindung zur Systemplatine (8085-Sockel)

(serial output data) beim 8085. Ihre Verwendung entscheidet, ob eine Adaptierung für Z80-Software möglich ist oder nicht.

Werden in einem 8085-System die Leitungen SID und SOD nicht verwendet, so bietet sich die Möglichkeit, durch Auskreuzen der Anschlüsse und Invertieren von sieben Leitungen einen NSC800 anstelle eines 8085 einzusetzen. Die entsprechende Schaltung ist in *Bild 1* dargestellt. *Bild 2* zeigt die praktische Realisierung der Schaltung in Form einer 70 × 30 mm² großen Adapterplatine, die oben zwei Sockel zur Aufnahme von NSC800 und Inverter (74LS240) aufweist und unten mit einem 40poligen Stecker versehen ist, der ein Einstekken in einen 8085-Sockel ermöglicht.

Dipl.-Ing. Franz Braunschmid

Bildschirmsteuerung mit EPROMs

Die Ansteuerung eines Bildschirms kann einen erheblichen Teil der Rechenzeit eines Mikrocomputers beanspruchen. Diese Zeit läßt sich auf ein Minimum beschränken, wenn man Information, die sich nicht ändert, in EPROMs ablegt. Eine solche Schaltung zeigt *Bild 1* (siehe nächste Seite). Das dargestellte Verfahren eignet sich für Anwendungen, in denen sich nur ein kleiner Teil des Bildschirminhalts ändert, während Linien, Symbole und Figuren als fester Hintergrund vorhanden sind.

Das System enthält RAM, EPROM und PROM. Diese Bausteine enthalten die Information, die der Y-Ablenkung (vertikal) entspricht. Im RAM werden die veränderlichen Daten abgelegt, der EPROM-Bereich enthält den festen Hintergrund, und im PROM sind Grafiksymbole und systemspezifische Zeichen gespeichert. Die X-Ablenkung (horizontal) wird von einem DA-Umsetzer gesteuert, der am Ausgang eines 10-Bit-Zählers liegt. Es handelt sich dabei um die oberen 10 Bits eines 13-Bit-Zählers. Die restlichen 3 Bits wählen den Baustein aus, der gerade für die Y-Ablenkung zuständig ist. Die Steuerlogik ist mit dem Mikroprozessor verbunden und liefert verschiedene Steuersignale, z. B. für die Speicher.

Greift der Prozessor nicht auf das System zu, wird der Takt zum 13-Bit-Zähler durchgeschaltet. Der Zähler adressiert der Reihe nach alle Speicherbausteine und schaltet dann zur nächsten Horizontalposition weiter, bis der gesamte Bildschirm beschrieben ist. Der Mikroprozessor hat über die Steuerlogik Zugriff auf das RAM und den voreinstellbaren Zähler.

Ein AD-Umsetzer stellt ein direktes Analog-Interface zum System dar. Damit läßt sich – wie mit einem Speicheroszilloskop – ein Analogsignal über längere Zeit speichern. Wenn das System im sogenannten Speicher-Modus ist, werden die RAM-Daten kontinuierlich auf dem Bildschirm dargestellt, und der Analogeingang ist nicht wirksam. Im Display-Modus werden die digitalisierten Analogwerte im RAM gespeichert und auf dem Bildschirm dargestellt. In diesem Fall ist eine 4-Kanal-Darstellung möglich. Mit einer zusätzlichen Triggerlogik könnten auch Teile des gespeicherten Kurvenzuges, ähnlich wie mit einem konventionellen Oszilloskop, herausgehoben werden. Im EPROM könnte man dabei Horizontal- und Vertikalachse sowie Zeitmarken ablegen. Numerische Informationen wie Wobbelfrequenz oder ähnliches könnten im PROM gespeichert sein.

Bild 2 zeigt die Blockschaltung eines Systems, bei dem sich die Intensität variieren läßt. Der Adreßzähler ist in diesem Fall auf 24 Bit erweitert. Wie vorher wählen die niederwertigen Bits den Speicherbaustein aus. Anstatt der Vertikalablenkung wird jetzt die Intensität des Elektronenstrahls in Abhängigkeit vom Speicherinhalt verändert. Die oberen Zählerbits sorgen wieder für die Horizontalablenkung. Von den verbleibenden Bits wird jede Speicherstelle adressiert, bevor der Strahl zur nächsten Horizontalposition wandert. Mit diesem Verfahren lassen sich fernsehbildähnliche Darstellungen verwirklichen.

John Rizzo

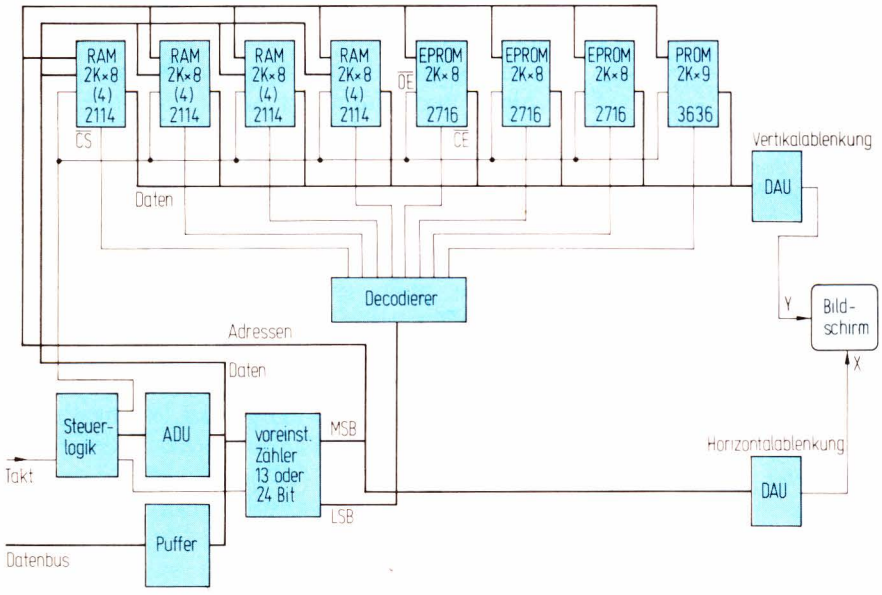


Bild 1. Schaltung für „digitale" Bilddarstellung

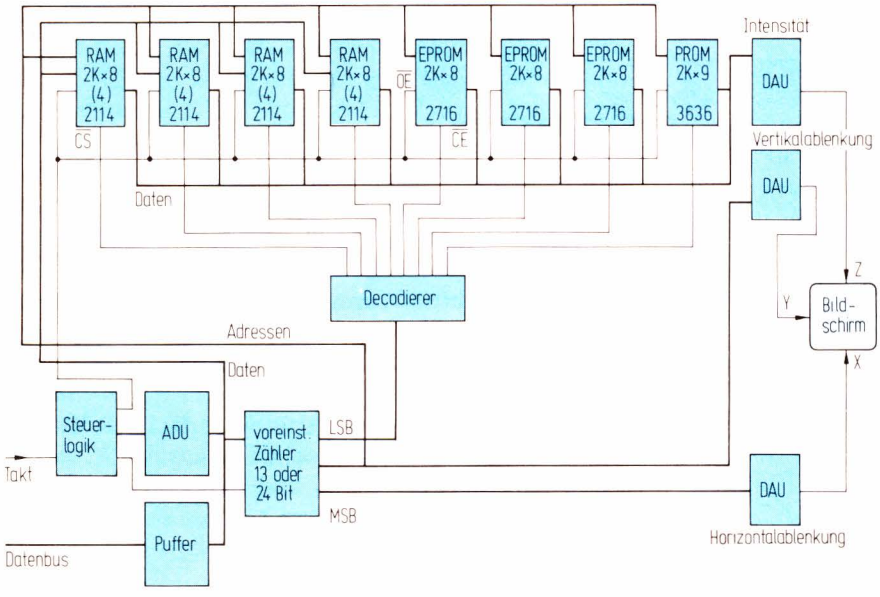


Bild 2. Schaltung für Bild mit Graustufen

Datenkommunikation
ein ELEKTRONIK-Sonderheft

Die Datenkommunikation hat sich innerhalb von ganz wenigen Jahren zu einer hochkomplexen Spezialwissenschaft entwickelt, angesiedelt zwischen EDV und Nachrichtentechnik. Für viele Ingenieure ist dieses junge Gebiet noch so etwas wie ein Buch mit sieben Siegeln. Dieses Heft leistet beim Einstieg wertvolle Hilfe. Es liegt jetzt in 2. erweiterter Auflage vor und enthält neben neuen Aufsätzen die in den letzten Jahren in der ELEKTRONIK zu diesem Thema erschienenen Beiträge.

Sonderheft: ELEKTRONIK-Arbeitsblätter

Damit Sie viele Grundlagen-Informationen schneller greifbar haben und nicht mehr aus vielen alten Heften mühsam manches zusammensuchen müssen, haben wir unsere bewährten Arbeitsblätter in diesem Sonderheft zusammengefaßt. Sie erhalten hiermit nun ein umfangreiches, firmenneutral gehaltenes Nachschlagewerk, das Ihnen über viele Jahre hinweg ein nützlicher Helfer sein wird. Die Themengruppen: Grundlagen, Analogtechnik, Digitaltechnik, Steuer- und Regeltechnik, Optoelektronik, Stromversorgung. Das Heft umfaßt 144 Seiten und kostet 21 DM. Bezugsmöglichkeiten überall dort, wo Sie die ELEKTRONIK erhalten – natürlich auch beim Franzis-Verlag selbst, Karlstr. 37, 8000 München 2, Tel. 0 89/51 17-1.

Analogsignalverarbeitung mit µC

Am Beispiel eines elektronischen Hallgerätes wird gezeigt, wie analoge Signale einfach mit einem µC (hier mit der CPU 8080) verarbeitet sowie ein- und ausgegeben werden können.

Zur Verarbeitung der Signale ist der Einsatz von A/D- und D/A-Umsetzern erforderlich (*Bild 1*). Ein preiswerter und schneller A/D-Umsetzer ist z. B. der Typ ZN 427E-8 der Firma Ferranti (maximale Übertragungsbandbreite ≈ 30 kHz, maximale Auflösung 128 Teile für jede Halbwelle (8 Bit)). Der Bandbreite sind noch durch das µC-System und durch die Länge des Programms Grenzen gesetzt. Es ist deshalb ein kleiner Hardware-Aufwand nötig, um den Vorgang zu beschleunigen. Der ADU benötigt einen Takt, der aus $\Phi_{2\ TTL}$ und mit Hilfe zweier Teilerstufen (FF 1 und FF 2) gewonnen wird (≈ 2 µs). Die Umsetzzeit beträgt neun Takte (18 µs). Der Startimpuls wird erst nach der Adressierung des ADU mit $\Phi_{2\ TTL}$ über FF 3 gebildet, damit das Einlesen der Daten gesichert ist. Ein Vorverstärker ist notwendig, um das Nf-Signal bis zu max. ±2,5 V für den ADU aufzubereiten.

Damit man den ADU optimal aussteuern kann, ist ein einfaches Peakmeter vorgesehen: Die digitalen Informationen des Analogwertes werden entsprechend verknüpft und einem Monoflop zugeführt. Im Fall der

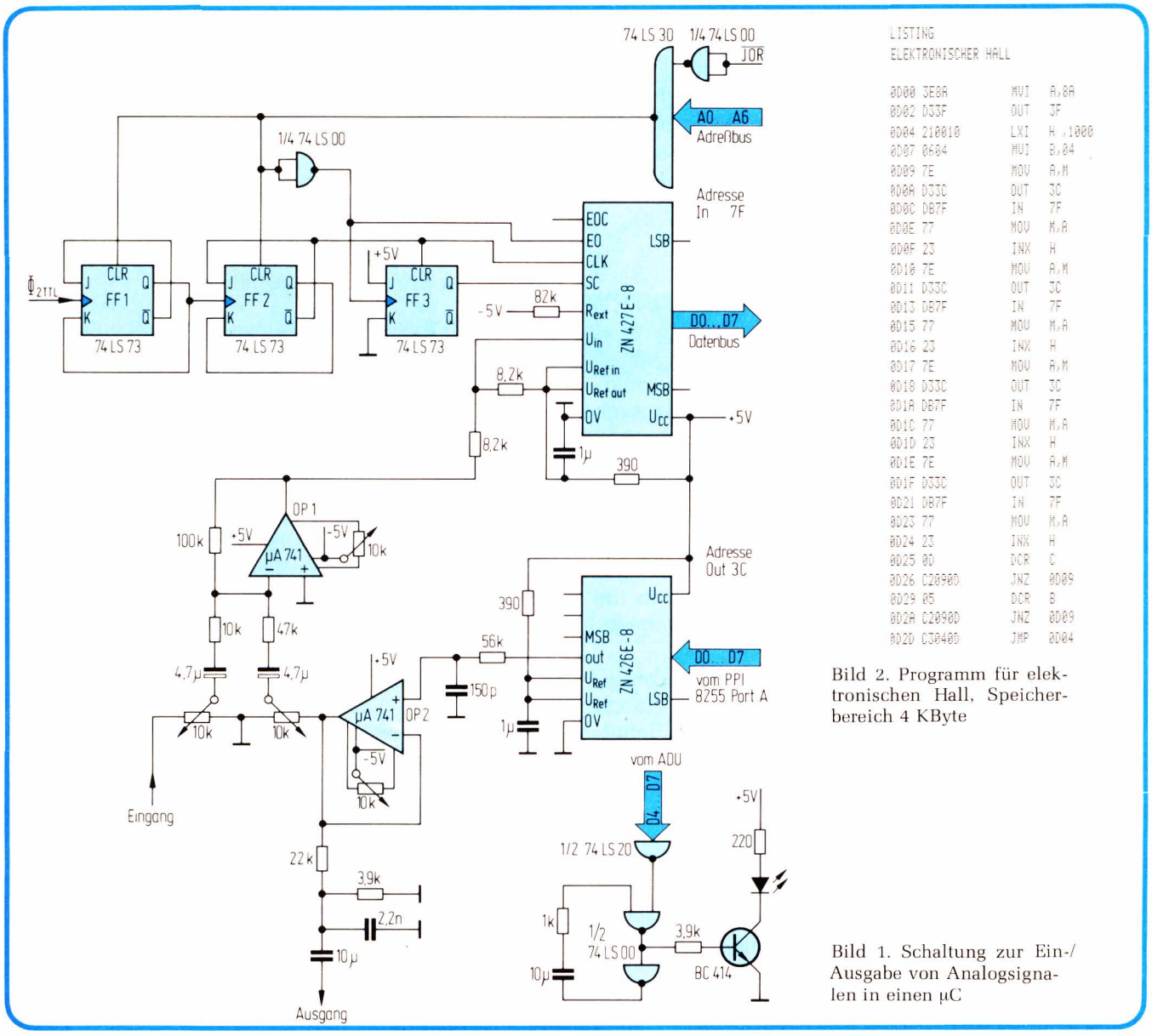


Bild 1. Schaltung zur Ein-/Ausgabe von Analogsignalen in einen µC

```
LISTING
ELEKTRONISCHER HALL

0D00 3E8A        MVI  A,8A
0D02 D33F        OUT  3F
0D04 210010      LXI  H,1000
0D07 0604        MVI  B,04
0D09 7E          MOV  A,M
0D0A D33C        OUT  3C
0D0C DB7F        IN   7F
0D0E 77          MOV  M,A
0D0F 23          INX  H
0D10 7E          MOV  A,M
0D11 D33C        OUT  3C
0D13 DB7F        IN   7F
0D15 77          MOV  M,A
0D16 23          INX  H
0D17 7E          MOV  A,M
0D18 D33C        OUT  3C
0D1A DB7F        IN   7F
0D1C 77          MOV  M,A
0D1D 23          INX  H
0D1E 7E          MOV  A,M
0D1F D33C        OUT  3C
0D21 DB7F        IN   7F
0D23 77          MOV  M,A
0D24 23          INX  H
0D25 0D          DCR  C
0D26 C20900      JNZ  0D09
0D29 05          DCR  B
0D2A C20900      JNZ  0D09
0D2D C30400      JMP  0D04
```

Bild 2. Programm für elektronischen Hall, Speicherbereich 4 KByte

nahezu vollständigen Aussteuerung leuchtet dann die LED (D 7 ∧ D 6 ∧ D 5 ∧ D 4 ≙ 240-128 ⇒ 112 Teile zu 128 Teile ≙ 87,5 %).

Als D/A-Umsetzer wird der Typ ZN 426E-8, ebenfalls von Ferranti, verwendet. Der Abschluß sollte hochohmig (≥ 600 kΩ) sein, damit die D/A-Umsetzung linear wird. Dies wird mit OP 2, der als Impedanzwandler geschaltet ist, erreicht.

Die Verzögerung des Nf-Signals erfolgt in der Weise, daß man es abtastet und den Digitalwert in den Arbeitsspeicher des µC schreibt (z. B. 4-KByte-Bereich). Anschließend wird der vorher eingelesene Wert wieder ausgelesen und in ein Nf-Signal umgesetzt. Die Verzögerungszeit ergibt sich dann aus der Differenzadresse, multipliziert mit der Abtastzykluszeit. Koppelt man das wiedergewonnene Nf-Signal zurück, erhält man einen elektronischen Hall (z. B. Schreibadresse – Leseadresse ≙ 4 KByte mal 20 µs ~ 90 ms). Das zugehörige Programm zeigt *Bild 2*.

Ing. (grad.) Johann Lentner

Time-Sharing-Betrieb für Mikroprozessoren

In Mikroprozessorsystemen wird viel Rechenzeit durch Warten auf langsame Peripherie vergeudet. Speziell in der Textverarbeitung nehmen die Ein-/Ausgabe-Operationen den größten Teil der Zeit in Anspruch. In diesem Beitrag wird ein aus Hardware und Software bestehendes Konzept erläutert, das mit nur einem Mikroprozessor bis zu fünfzehn Bildschirmterminals bedienen kann.

In einem Textverarbeitungssystem würde sich die Kombination von 13 Bildschirmgeräten und zwei Druckern mit unterschiedlichen Schriftarten anbieten. Gute Drucker erfordern gegenüber Bildschirmterminals einen hohen Investitionsaufwand, dagegen wird die Kapazität des Druckers in einem Einzelsystem nur zu einem geringen Prozentsatz genutzt. Mit diesem Konzept kann das Verhältnis der Anzahl an Druckern zur Anzahl der Bildschirmgeräte optimiert werden und so zu erheblichen Einsparungen führen. Die Texteingabe und Korrektur erfolgt am Bildschirm, und der endgültige Text wird dann in die Ausgabeschlange des jeweiligen Druckers eingereiht.

Das Konzept basiert auf dem Time-Sharing-/Multi-User-Prinzip. Jedem Benutzer wird für eine kurze Zeitspanne der Prozessor zugeteilt. Wenn das Benutzerprogramm den Zugriff auf eine Peripherieeinheit abwarten muß, wird es dem nächsten Benutzer zugeteilt. In diesem Fall ist die Zuteilungszeit auf eine Millisekunde eingestellt, da dies das Optimum zwischen Verarbeitungsgeschwindigkeit und der Reaktionszeit auf Terminalbedienungen darstellt. Über einen Wahlschalter können auch folgende andere Zeiten eingestellt werden: 0,5 ms, 2 ms, 4 ms, 8 ms.

Damit keine Interaktionen zwischen den einzelnen Benutzerprogrammen entstehen können, wird jedem Benutzer ein eigener Speicherblock zugeordnet. Die einzelnen Speicherblöcke schaltet eine spezielle Schaltung um (*Bild 1*). Sollte dieses Umschalten mit Software bewerkstelligt werden, so nähme ein solches Programm nicht mehr zu vernachlässigende Zeit in Anspruch, die die Geschwindigkeit des Systems erheblich reduzieren würde.

In diesem Beispiel wird der Prozessor MC6800 mit einer Taktfrequenz von 2 MHz benutzt. Die im folgenden beschriebene Hard- und Software ist aber mit nur geringen Modifikationen auch auf anderen stackorientierten Prozessoren lauffähig.

Da dieses System interruptgesteuert ist, müssen die Ein-/Ausgabe-Programme geringfügig geändert werden. Im Prinzip arbeiten alle diese Unterprogramme gleich. Hier sollen nur die E/A-Routinen für eine serielle Schnittstelle betrachtet werden (*Bild 2*). Zuerst überprüft der Prozessor, ob die Schnittstelle bereit ist, einen Datentransfer zu übernehmen. Ist ein vorangegangener Transfer noch nicht abgeschlossen, so wird über den Lesebefehl LDAA INT ein Interrupt erzeugt und das E/A-Programm unterbrochen. Im Interrupt-Dienstprogramm wird dem nächsten Benutzer der Zugriff gegeben. Ist dagegen ein Datentransfer möglich, so wird dieser eingeleitet und anschließend in das aufrufende Programm zurückgekehrt.

Nach Ablauf der zugeteilten Zeit unterbricht ein Non-Maskable-Interrupt das laufende Benutzerprogramm und springt nach dem Retten der Registerinhalte in die Interrupt-Routine. Dort wird der augenblickliche Inhalt des Stackpointers in die Speicherzelle „STACK" gerettet. Die Logik schaltet durch den Befehl STAA INT auf den nächsten Speicherblock um. Da durch diese Umschaltung auch die Adresse STACK mit umgeschaltet wird, kann daraus der neue Stackpointer-Inhalt geladen werden und das nun aktive Programm, nach Rückkehr aus der Interrupt-Routine, fortgesetzt werden.

Das Programm USEROF gibt zugeteilte Rechenzeit sofort an den nächsten Benutzer weiter, indem es über LDAA INT einen Interrrupt erzeugt. Es wird nur durchlaufen, wenn das zugehörige Terminal ausgeschaltet ist. Dadurch erhöht sich die Zuteilungshäufigkeit der anderen Terminals, und der Prozessor wird nicht unnötig blockiert. USEROF kann nur durch selektiven Reset oder Gesamt-Reset verlassen werden. Die Reset-Routine initialisiert den Stackpointer und gibt sofort den Zugriff für den nächsten Benutzer frei, um dessen Aktivitäten nicht zu verzögern.

Alle diese Programme sind in jedem Speicherblock vorhanden und befinden sich unter identischen

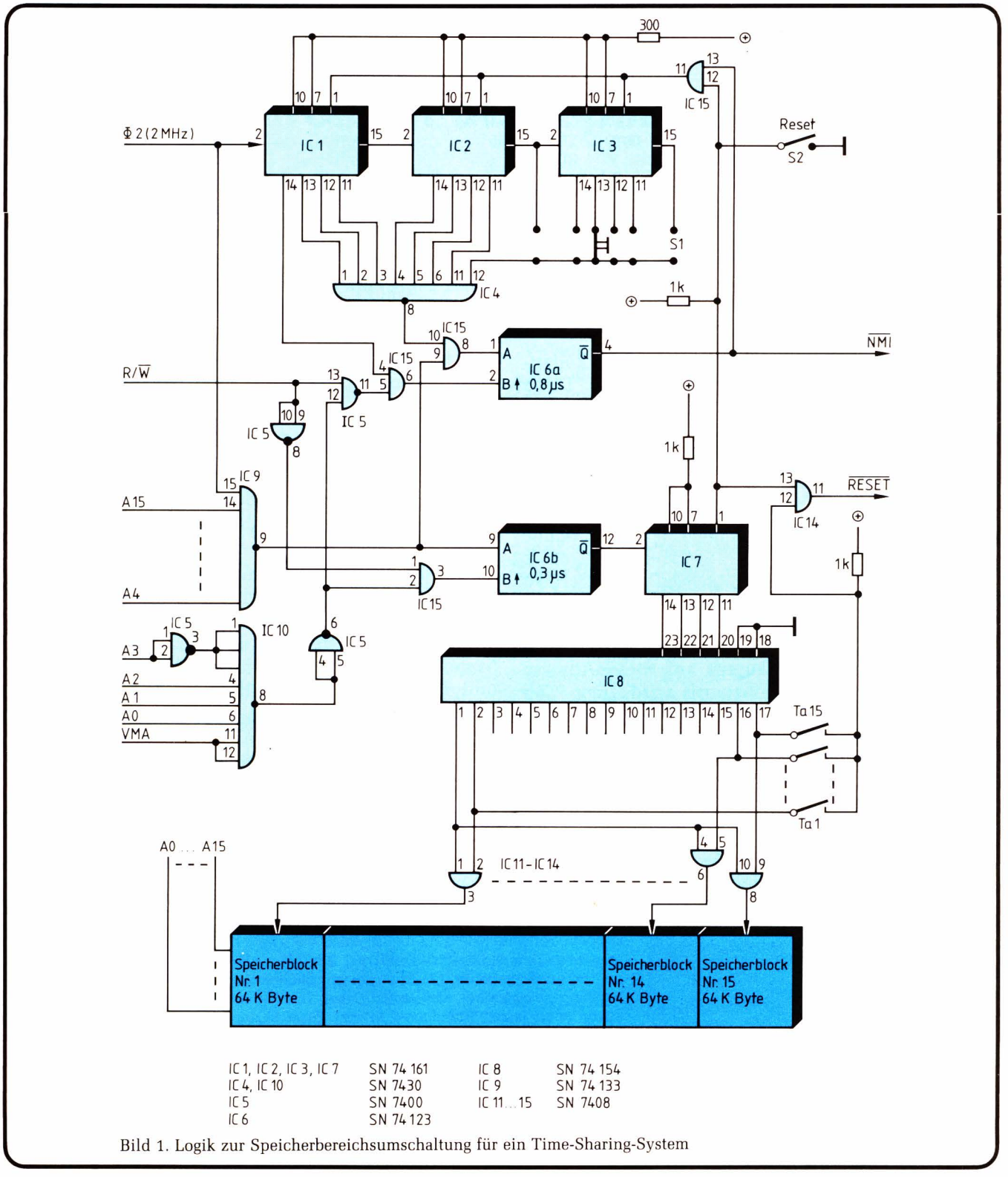


Bild 1. Logik zur Speicherbereichsumschaltung für ein Time-Sharing-System

```
        ORG  $A000
STACK   EQU  $FFF5
INT     EQU  $FFF7
ACIAD   EQU  CF7E
ACIAC   EQU  CF7F

Interrupt-Dienstprogramm

INTER   STS  STACK
        STAA INT
        LDS  STACK
        RTI

Zeitweitergabe bei abgeschaltetem Terminal

USER OF LDAA INT
        BRA  USER OF
```

```
Einlesen eines Zeichens (ASCII)

INCH    LDAA ACIAC
        ASRA
        BCS  INCH 1
        LDAA INT
        BRA  INCH
INCH 1  LDAA ACIAD
        RTS

Ausgabe eines Zeichens (ASCII)

OUTCH   LDAB ACIAC
        ASRA
        ASRA
        BCS  OUTCH 1
        LDAA INT
        BRA  OUTCH
OUTCH1  STAA ACIAD
        RTS
```

```
Reset-Routine (nur Initialisierung des SP)

SPINIT  FDB  $7FFF
RESET   LDS# SPINIT
        STAA INT
          ¦    ¦
          ¦    ¦
```

Bild 2. Programme für den Time-Sharing-Betrieb am Beispiel einer seriellen Schnittstelle

Adressen. Genauso muß der Stack-Bereich in allen Blöcken bei der gleichen Adresse beginnen.

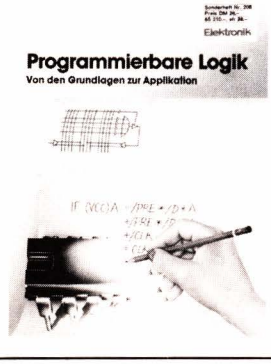
Aktive Programme werden nach Ablauf der Zeitzuteilung durch NMI-Impulse unterbrochen. Diese Interrupt-Impulse werden aus dem Systemtakt des Mikroprozessors durch Teilung der Taktfrequenz gewonnen. Das NAND-Gatter IC4 erzeugt den Teilungsfaktor aus den Ausgängen der drei 4-Bit-Zähler IC1...3. Mit S1 kann der Teilungsfaktor eingestellt werden. IC4 triggert über das Monoflop IC6a den NMI-Eingang des Mikroprozessors und setzt die Zähler auf Null. IC6a hält NMI für etwa 0,8 µs auf Low, um eine sichere Übernahme durch den Prozessor zu gewährleisten.

IC5, IC9, IC10 und IC15 decodieren die Adresse FFF7 und erzeugen bei Zugriff auf diese Adresse einen Impuls. Beim Schreibzugriff wird, über IC6b verzögert, der Dualzähler IC7 um eins erhöht. Die Ausgänge des Zählers selektieren über den BCD/Dezimal-Decodierer IC8 einen der 15 möglichen Speicherblöcke. Ein Lesezugriff auf FFF7 erzeugt mit IC6a einen Interrupt. So kann mit LDAA INT das laufende Programm softwareseitig unterbrochen werden.

Zur Initialisierung des Gesamtsystems kann durch Schließen von S2 das ganze System zurückgesetzt werden. Sämtliche Zähler werden auf Null gesetzt, und Ausgang Q1 von IC8 selektiert alle Speicherblöcke gleichzeitig und initialisiert den Stackpointer. Der selektive Reset wird vom jeweiligen Benutzer mit der ihm zugeordneten Taste Ta ausgelöst und betrifft nur sein Programm. Denn nur, wenn der ihm zugeordnete Speicherblock selektiert ist, aktiviert der BCD/Dezimal-Decodierer die entsprechende Reset-Taste.

An diesem Konzept kann man erkennen, daß eine sinnvolle Kombination von Hard- und Software die Leistungsfähigkeit eines Prozessors beträchtlich erhöhen kann. Ein Problem, das sich bei diesem System ergeben kann, ist der Ausfall des Prozessors und damit die Stillegung aller angeschlossenen Peripherie. Durch Parallelschalten von zwei Prozessoren kann bei Ausfall des einen der andere dessen Peripheriegeräte mitbedienen. So erhält man ein System, dessen Kapazität optimal genutzt werden kann und das hohe Betriebssicherheit gewährleistet.

Helmut Höper



Sonderheft: Programmierbare Logik

Was mit Software nicht und mit Standard-Hardware nur mit unvertretbarem Aufwand realisiert ist, das genau läßt sich mit programmierbaren Logikbausteinen ökonomisch und technisch einwandfrei realisieren. Diese Technik hat sich schon ihren festen Standpunkt in den Entwicklungslabors verschafft und wird ihn in den nächsten Jahren noch ausbauen. Einen Eindruck davon, welche Möglichkeiten in der programmierbaren Logik stecken, soll Ihnen dieses Sonderheft vermitteln. Es umfaßt 96 Seiten, kostet 28 DM und ist überall dort erhältlich, wo Sie die ELEKTRONIK bekommen, außerdem natürlich auch beim Franzis-Verlag, Karlstr. 37, 8000 München 2, Tel. 089/51 17-1.

Datenerfassung mit 80186:

Meßwerte direkt in den Speicher gebracht

Für die Meßwerterfassung mit Mikroprozessorsystemen kommen verschiedene Möglichkeiten in Frage. Steht wenig Zeit zur Verfügung, dann bietet sich das Verfahren des direkten Speicherzugriffs (DMA, Direct Memory Access) an. Besonders elegant läßt sich das Problem mit dem Prozessor 80186 lösen, der über zwei DMA-Kanäle verfügt.

Computergesteuerte Meßwerterfassungssysteme müssen im allgemeinen drei Aufgaben erledigen:
- die Meßwerte erfassen (z. B. mit einem A/D-Umsetzer)
- die Daten in bestimmte Größen umrechnen (z. B. in mV)
- die berechneten Werte in eine dem Menschen verständliche Form bringen (z. B. grafisch darstellen)

Soll eine Reihe von Messungen im Millisekundenbereich durchgeführt werden, dann bieten sich mehrere Möglichkeiten an.

1 Synchrone Meßwerterfassung

Bei der synchronen Meßwerterfassung wird der Meßzeitpunkt bzw. der Zeitablauf der Messungen vom Prozessor festgelegt. Es wird jeweils nacheinander ein Meßwert erfaßt, umgerechnet und dargestellt, bevor die nächste Messung beginnt. Dieses Verfahren funktioniert jedoch nur dann, wenn die Meßdaten zum Zeitpunkt des Einlesens durch die CPU auch wirklich vorhanden sind. Ein Zwischenspeichern der gesamten Meßreihe im RAM ist nicht notwendig, da jeder erfaßte Meßwert sofort weiterverarbeitet wird.

2 Asynchrone Meßwerterfassung

Bei der asynchronen Meßwerterfassung gibt es zwei Möglichkeiten: interruptgesteuerte oder DMA-gesteuerte Meßwerterfassung. Bei der interruptgesteuerten Meßwerterfassung erreichen die Meßanforderungen die CPU zu beliebigen Zeiten über eine Interrupt-Request-Leitung. Bei jeder Interruptanforderung springt die CPU in eine Interruptserviceroutine, speichert einen Meßwert im RAM und setzt daraufhin das normale Programm fort.

Das Hauptprogramm verarbeitet dabei unabhängig davon, in welche Speicherzelle die Interruptroutine den gerade anfallenden Meßwert schreibt, der Reihe nach die im RAM gespeicherten Daten. Der Nachteil dieses Verfahrens ist die bei steigender Zahl von Meßwerten länger werdende Zeit für die Interruptbehandlung. Falls die Verarbeitungsgeschwindigkeit der CPU höher ist als die Folge der ankommenden Meßwerte, ist eine Zwischenspeicherung im RAM nicht notwendig, da die Meßwerte direkt nach dem Interrupt verarbeitet werden können und die CPU bis zum nächsten Interrupt in einer Warteschleife verweilen kann.

Bei der DMA-gesteuerten Meßwerterfassung werden die Meßdaten zu beliebigen Zeiten, die von externer Seite vorgegeben werden, direkt in einen vorher festgelegten Speicherbereich übertragen. Da jeder DMA-Zyklus nur zwei Prozessorzyklen – einen READ-Zyklus und einen WRITE-Zyklus – benötigt, wird die Rechengeschwindigkeit des Prozessors nur unerheblich vermindert. Der entscheidende Vorteil jedoch ist, daß die Software zum eigentlichen Messen überhaupt nicht in Anspruch genommen wird. Unabhängig vom Einschreibvorgang können die Daten nun von der CPU weiterverarbeitet und dargestellt werden. Zu beachten ist, daß bei dieser Art der Meßwertverarbeitung kein zeitlicher Zusammenhang zwischen Meßwerterfassung und -darstellung existiert.

3 Anwendungsbeispiel

Im folgenden wird ein asynchrones, DMA-gesteuertes Meßwerterfassungssystem vorgestellt, das mit der CPU 80186, einem AD-Umsetzer, einer seriellen Schnittstelle sowie RAM und EPROM realisiert wurde. Die Aufgabe bestand darin, während eines sich regelmäßig wiederholenden Sägezahnsignales von 100 ms Dauer, das von

DMA-Steuerblock

Registername	Registeradresse	
	Kanal 0	Kanal 1
Source-Pointer	FFC0H	FFD0H
Source-Pointer (obere 4 Bit)	FFC2H	FFD2H
Destination-Pointer	FFC4H	FFD4H
Destination-Pointer (obere 4 Bit)	FFC6H	FFD6H
Transfer-Count	FFC8H	FFD8H
Steuerwort	FFCAH	FFDAH

Bild 1. Die Funktion der DMA-Kanäle wird beim 80186 von jeweils sechs Registerpaaren festgelegt

einem Funktionsgenerator mit Pausen von 10 ms erzeugt wurde, 256 Meßwerte aufzunehmen, umzurechnen und bei Bedarf die Mittelwerte von jeweils zehn Messungen über die serielle Schnittstelle auszugeben.

3.1 Hardware

Als Prozessor für dieses System wurde die CPU 80186 ausgewählt, die über zwei DMA-Kanäle auf dem Chip verfügt. Die Funktion der DMA-Kanäle wird durch den Inhalt von jeweils sechs 16-Bit-Registern festgelegt, die sich im Steuerblock der CPU befinden (*Bild 1*). Sie enthalten einen 20-Bit-Source-Pointer, einen 20-Bit-Destination-Pointer, ein Register für die Anzahl der auszuführenden DMA-Zyklen sowie das DMA-Steuerwort.

Als Speicher stehen 64 KByte EPROM und 16 KByte statisches RAM, aufgeteilt in Low- und High-Byte-Bereich, zur Verfügung (*Bild 2*). Der EPROM-Bereich, festgelegt durch die Programmierung des UMCS-Registers im Prozessor, erstreckt sich von F0000H...FFFFFH. Der RAM-Bereich, in dem sich auch die Interruptvektoren befinden, umfaßt den Bereich von 0H...3FFFH, festgelegt durch die Programmierung des LCS-Registers. Die Adressen ADR0...ADR15 werden mit zwei Latches aus den Multiplex-Adreß-/Datenleitungen AD0...AD15 erzeugt. Der serielle Schnittstellenbaustein 8251 und der AD-Umsetzer AD574 werden über zwei Datenpuffer abgetrennt, um einen Buskonflikt mit RAMs und EPROMs zu vermeiden. Ihre Richtung wird direkt vom DT/$\overline{R}$-Signal des 80186 gesteuert, ihr Enable-Signal entsteht aus einer Verknüpfung von $\overline{PCS0}$, $\overline{PCS1}$ und DEN des 80186. Die für den 8251 nötigen Taktsignale erzeugen die programmierbaren Zeitgeber des 80186. Timer 0 liefert das Taktsignal von 2 MHz, Timer 1 mit einem Teilungsfaktor von 26 die Frequenz 76,8 kHz, aus der der 8251 im 1/16-Mode die serielle Schnittstelle (4800 Bd) bedient.

Der Meßwerterfassungsteil des Prozessorsystems besteht aus einer Logik, die pro Sägezahnsignal jeweils

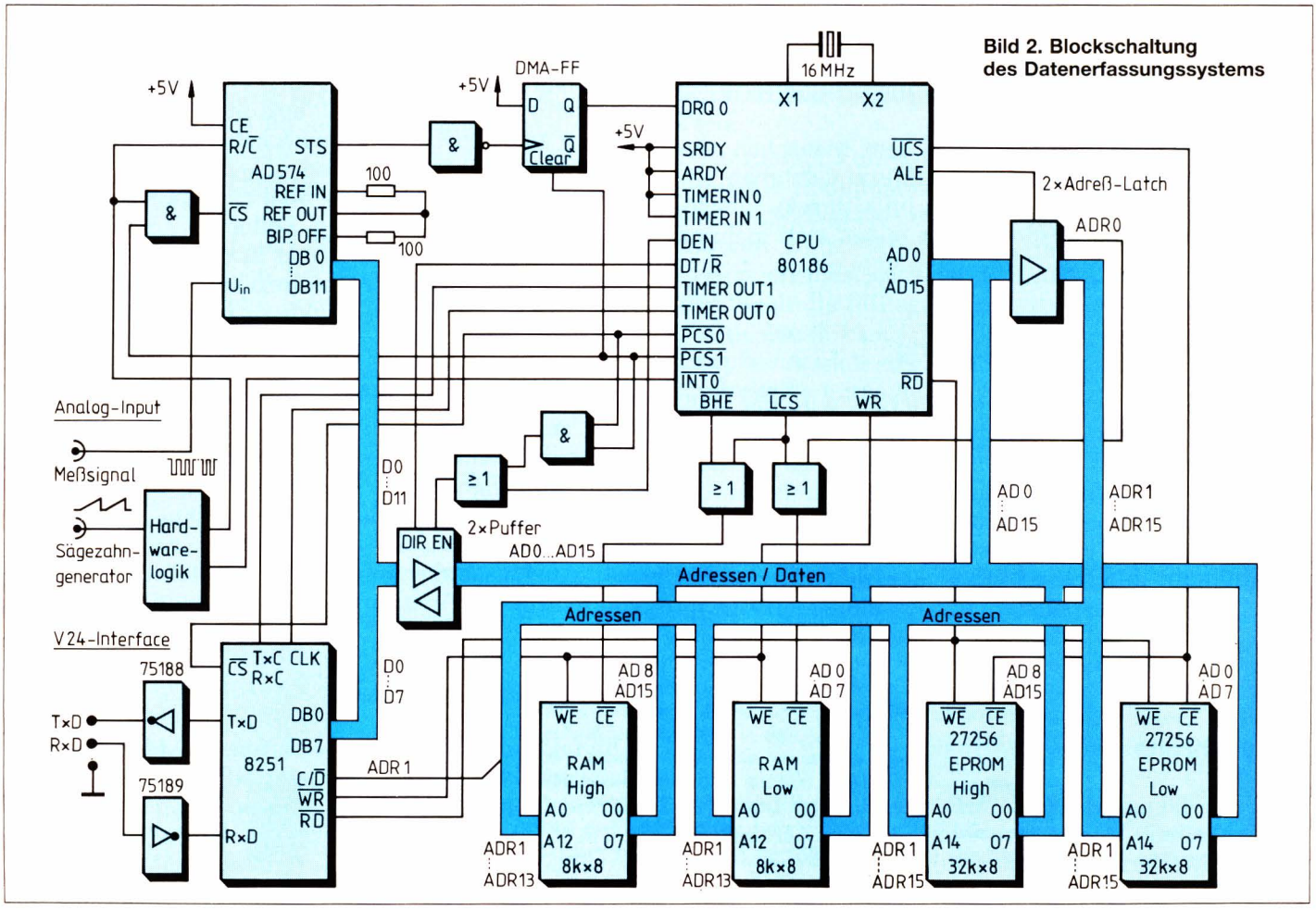


Bild 2. Blockschaltung des Datenerfassungssystems

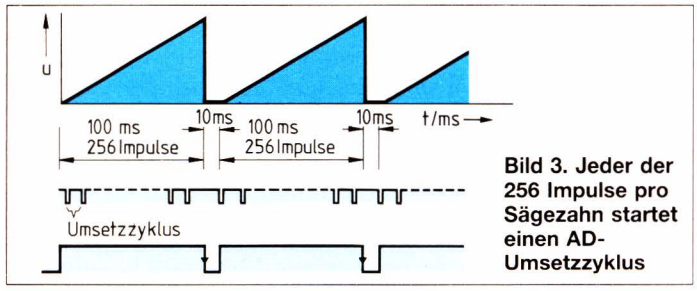


Bild 3. Jeder der 256 Impulse pro Sägezahn startet einen AD-Umsetzzyklus

256 negative Impulse von 500 ns Dauer im Abstand von 400 µs erzeugt. Sie starten einen Umsetzzyklus des AD574 (*Bild 3*). Nach beendeter AD-Umsetzung wird über den STS-Ausgang des AD574 ein DMA-Flipflop gesetzt, das eine DMA-Anfrage beim 80186 auslöst. Dieser holt daraufhin im nächsten Prozessorzyklus das Ergebnis der Umsetzung von der Adresse 80H ($\overline{PCS1}$ = 80H...FFH) ab und schreibt es im nächsten Maschinenzyklus in den durch die Programmierung des DMA-Destination-Registers festgelegten Speicherbereich von 1000H...11FFH. Gleichzeitig wird durch das $\overline{PCS1}$-Signal das DMA-Flipflop und damit das DRQ0-Signal des 80186 zurückgesetzt, so daß kein weiterer DMA-Zyklus ausgeführt wird.

Beim nächsten negativen Impuls wird der A/D-Umsetzer erneut gestartet, und der ganze Ablauf beginnt von neuem. Da der DMA-Controller im 80186 keine Autoload-Fähigkeit besitzt, wird nach 256 Impulsen bzw. einem Vorlauf des Sägezahnes von der Logik ein Interrupt (INT0) erzeugt, um ein erneutes Laden des DMA-Count-Registers zu ermöglichen.

3.2 Software

Eine Speicherübersicht des gesamten Systems zeigt *Bild 4*. Da die CPU Register enthält, die die Adreßbereiche von Speicherblöcken und E/A-Bereichen festlegen, müssen diese vor dem Start des eigentlichen Programmes in einem separaten Initialisierungsmodul programmiert werden (INIT-186).

Auf der Startadresse FFFF0H des 80186 ist ein „FAR-JUMP" auf dieses Initialisierungsmodul abgelegt. Da nach dem Reset nur der oberste 1-KByte-Bereich des Speichers adressiert werden kann, muß sich dort auch das Initialisierungsmodul befinden. In ihm werden sämtliche internen Register des 80186, die für die Adressierung von EPROM, RAM und Peripherie zuständig sind, auf die entsprechenden Werte gesetzt. Nach dem Durchlaufen der Initialisierung erfolgt ein Sprung in den Debug-Monitor, von welchem aus nun das eigentliche Meßprogramm gestartet werden kann.

Am Beginn dieses Programmteiles wird der DMA-Steuerblock des 80186 mit Quelladresse, Zieladresse, Anzahl der DMA-Zyklen sowie dem DMA-Steuerwort geladen (*Bild 5*).

Das eigentliche Meßwertverarbeitungsprogramm besteht aus einer Schleife, die kontinuierlich alle 256 Werte aus dem RAM-Bereich 1000H...11FFH umrechnet und bei Bedarf über die serielle Schnittstelle ausgibt. Unterbrochen wird dieser Ablauf nur durch den Sprung in eine Interruptserviceroutine am Ende jedes Sägezahnvorlaufes, in der dann das DMA-Count-Register wieder neu geladen wird.

Dipl.-Ing. Thomas Criegee

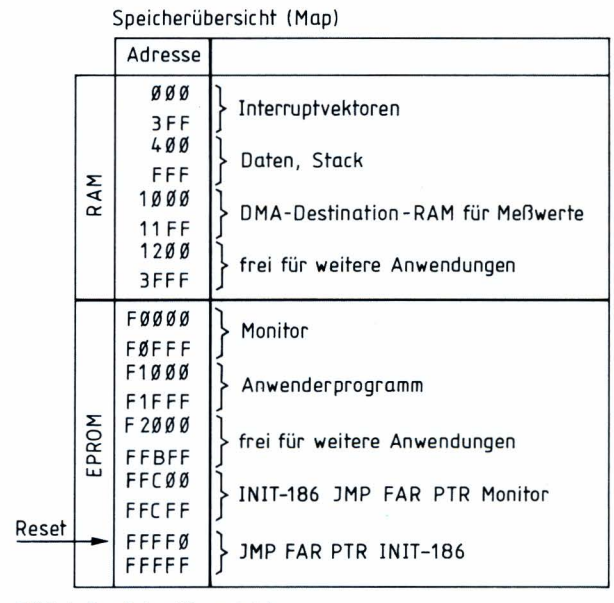


Bild 4. Speicherübersicht

```
;*********************************************
;*                                           *
;*        DMA SETUP   CHANNEL 0              *
;*                                           *
;*********************************************
MOV     DX,0FFC4H        ;DEST. POINTER LOW
MOV     AX,1000H
OUT     DX,AX
MOV     DX,0FFC6H        ;DEST. POINTER HIGH
MOV     AX,0H
OUT     DX,AX
MOV     DX,0FFC0H        ;SOURCE POINTER LOW
MOV     AX,80H
OUT     DX,AX
MOV     DX,0FFC2H
MOV     AX,0H            ;SOURCE POINTER HIGH
OUT     DX,AX
MOV     DX,0FFC8H        ;TRANSFER COUNT
MOV     AX,256D
OUT     DX,AX
MOV     DX,0FFCAH        ;CONTROLWORD
MOV     AX,0A265H
OUT     DX,AX
```

Bild 5. So lädt man den DMA-Steuerblock mit den erforderlichen Parametern

Selbstbau oder Kauf?

Eine CPU-Platine mit dem iAPX 286 muß nicht teuer sein

Am Institut für Physiologie der FU Berlin sollte ein Meßdatenerfassungssystem mit 16 Analog- und Ereigniskanälen und einer Abtastfrequenz von 1 kHz entwickelt werden, das die Daten vorverarbeitet und dann an einen übergeordneten Rechner überträgt. Durch Verzicht auf einige zwar wünschenswerte, aber für die Anwendung im Institut entbehrliche Eigenschaften von teuren „käuflichen" Platinen war es möglich, durch Selbstbau eine preislich akzeptable Lösung zu finden.

Stößt man bei der Erfassung von Meßdaten und deren Vorverarbeitung in Echtzeit an die Grenzen der Leistungsfähigkeit des Rechners, so gibt es zwei Möglichkeiten: Man kann eine schnellere Zentraleinheit (CPU) verwenden oder durch Einsatz von weiteren Prozessoren die Intelligenz im System verteilen. Die erste Möglichkeit ist durch den Ersatz der vorhandenen CPU-Platine verhältnismäßig leicht zu bewerkstelligen, während die zweite Möglichkeit umfangreiche Änderungen in der Hardware und eine grundsätzlich andere Organisation der Software nach sich zieht.

Die Aufgabe bestand darin, unter Aufrechterhaltung aller sonstigen Eigenschaften, wie Belegung der Speicherbereiche und Adressierung der Ein-/Ausgabebausteine, die bereits vorhandene CPU-Karte (mit einem 8086) mit dem 80286 auszurüsten, und das ohne wesentliche Softwareänderungen. Dieses Vorhaben wird durch die Betriebsart „Real Address Mode" des iAPX 286 ermöglicht, in der man bewußt auf einige hervorragende Eigenschaften wie Zugriffsschutz und virtuelle Adressierung verzichtet, dafür aber die vorhandenen Programme des iAPX 86 fast unverändert einsetzen kann.

Die CPU 80286 arbeitet mit dem Buscontroller 82288 und dem Clockgenerator 82284 zusammen (*Bild 1*). Der Taktgenerator erzeugt das 12-MHz-CLK-Signal, das in der CPU halbiert wird und den Prozessortakt PCLK bildet. Er liefert außerdem das READY-Signal, das wahlweise aus einem synchronen (SRDY) oder asynchronen

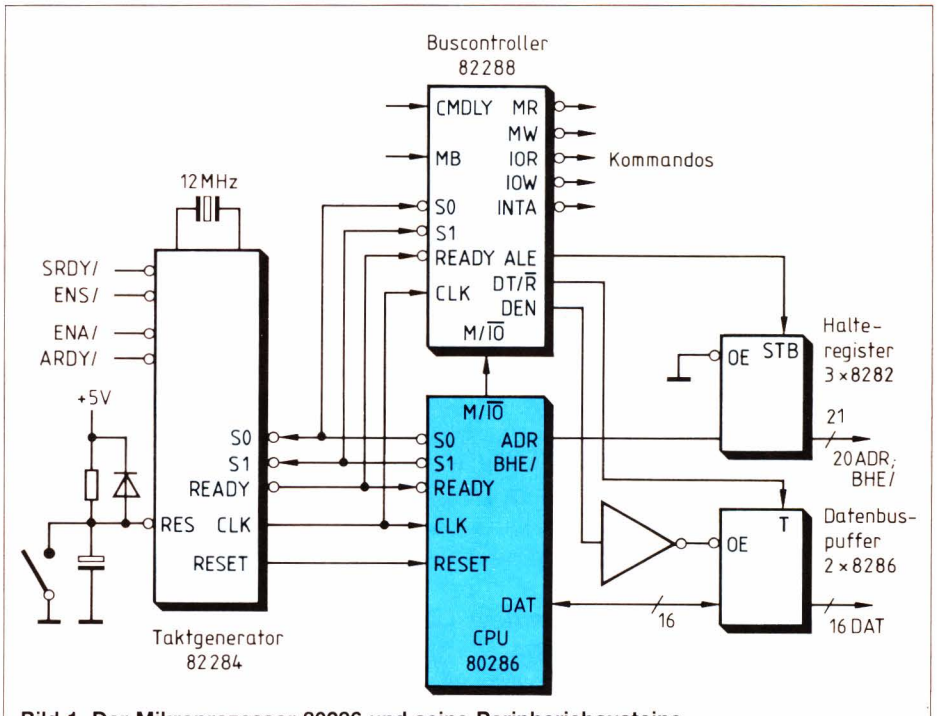


Bild 1. Der Mikroprozessor 80286 und seine Peripheriebausteine

Eingang (ARDY) stammen kann, je nachdem, ob es auf der eigenen Platine oder über den Systembus, der dem Multibus ähnlich ist, von anderen Baugruppen erzeugt wird. Der Buscontroller erhält von der CPU die Statussignale S0, S1 und M/IO und erzeugt daraus die entsprechenden Kommandos für das Lesen und Schreiben der Speicher- und E/A-Bausteine. Außerdem steuert er die Richtung und Einschaltdauer der Datenbuspuffer mit den Signalen DT/R und DEN und bestimmt mit dem ALE-Signal den Zeitpunkt, zu dem die Adressen in die Halteregister übernommen werden.
Der Buscontroller kann sehr vielseitig an die zeitlichen Erfordernisse der verwendeten Peripherie angepaßt werden. Mit Hilfe des Einganges CMDLY (Commanddelay) kann das Erscheinen der Kommandos in bezug auf die Adreß- und Selekt-Signale verzögert werden, während durch READY die Kommandos in ihrer Zeitdauer verändert werden können. Durch den Eingang MB (Multibus) werden die Kommandos an den Multibus angepaßt, wodurch eine verzögerte Ausgabe der Schreib- und Lesesignale bewirkt wird.

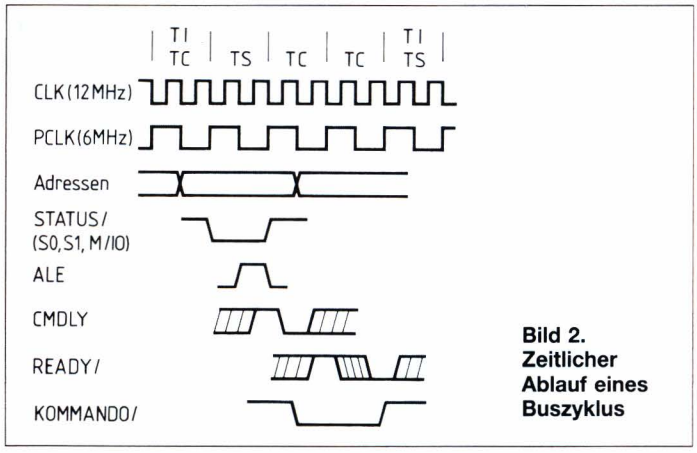


Bild 2. Zeitlicher Ablauf eines Buszyklus

Zeitliche Abläufe

Die CPU beginnt jeden Buszyklus mit der Phase TS (*Bild 2*), in der sie den Buscontroller über die Art des zu erwartenden Kommandos informiert. Sie hat die Adressen für einen gerade laufenden Buszyklus bereits während des letzten Teiles des vorangegangenen bereitgestellt. Das hat den Vorteil, daß die Selektsignale für die Bausteine auf der eigenen Platine im voraus erzeugt werden können und die Zeit zur Decodierung der Adressen nicht verzögernd wirkt. Andererseits müssen aber die Adreß- und Selektsignale durch ALE, das im zweiten Teil von TS erscheint, in Halteregister übernommen werden, damit sie während des ganzen Buszyklus konstant bleiben.

Auf TS folgt die Kommandophase TC, in der der Buscontroller die Schreib- oder Lesesignale ausgibt. Dabei berücksichtigt er das Signal CMDLY, um evtl. den Einsatz der Kommandos zu verzögern. Die TC-Phasen werden so oft wiederholt (Waitstates), bis am Ende einer Phase das READY-Signal aktiv wird. Wenn der Prozessor fortlaufend Daten transportiert, wie es bei den MOVE-String-Befehlen der Fall ist, kann auf TC sofort wieder TS folgen. Andernfalls werden TI-Phasen eingefügt.

Buscontroller und CS-Decoder

Um den zeitlichen Ablauf für Zugriffe auf Bauelemente auf der Platine (lokal) oder über den Systembus zu optimieren, werden zwei Buscontroller eingesetzt (*Bild 3*). Mit Hilfe des CS-Decoders wird ein Signal LOCAL erzeugt, das jeweils für einen Buszyklus einen der beiden Controller aktiviert und auch die richtige Quelle für das READY-Signal am Taktgenerator auswählt.

Der Buscontroller für den Systembus arbeitet in der Betriebsart Multibus (MB=1). Er verzögert den Einsatz aller Kommandos um eine Clockperiode. Die Länge des Kommandos richtet sich nach dem Erscheinen des Signales XACK, das von der jeweils angesprochenen Platine im System erzeugt werden muß.

Der Buscontroller für lokale Zugriffe arbeitet in der Betriebsart MB=0. Für die Zusammenarbeit der CPU mit den Speicher- und Ein-/Ausgabebausteinen muß der Zeitbedarf für jeden Baustein berücksichtigt werden, um entscheiden zu können, wieviel die Kommandos mit Hilfe des Einganges CMDLY verzögert werden und auf welche Länge sie mit Hilfe des SRDY-Einganges gestreckt werden müssen. Deshalb teilt der CS-Decoder dem Wait-State-Generator mit den Signalen S0WAIT bis S2WAIT für jeden lokalen Zugriff die Anzahl der benötigten Wait-States mit und steuert mit SCMDLY den Eingang CMDLY am lokalen Buscontroller.

Der CS-Decoder erzeugt die Chip-Selekt-Signale für alle lokalen Bausteine. Er besteht aus drei PAL-Bausteinen (Programmable Array Logik). Seine Ausgänge werden durch zwei transparente Halteregister geleitet, in denen sie mit dem ALE-Signal während eines Zugriffes „eingefroren" werden. Da auch die Adreßpuffer mit ALE gesteuert werden, erscheinen Adressen und Selekt-Signale zum gleichen Zeitpunkt an den Bausteinen.

Beim Entwurf des CS-Decoders ist darauf zu achten, daß die CPU auch Bytes aus dem PROM lesen kann. Beim iAPX 86 war das nicht notwendig, weil die CPU die Instruktionen immer in 16-Bit-Worten las. Beim iAPX 286 kann es vorkommen, daß die CPU nur das obere Byte liest, um den Zugriffsschutz nicht zu verletzen. Dieser Fall tritt auch im „Real Address Mode" ein, wenn ein Sprungbefehl eine ungerade Adresse zum Ziel hat.

Beim Auftreten von Unterbrechungen aktiviert der PIC 8259A den INT-Eingang der CPU, worauf diese mit Hilfe des INTA-Signales eine Kennzahl aus dem PIC liest, mit dem sie einen Zeiger auf den dazugehörenden Interruptvektor bildet. Dieser Vorgang ist ein lokaler Zugriff, der vom Decoder nicht durch Beobachtung der Adressen, sondern nur durch den Status (S0, S1, M/IO) erkannt werden kann.

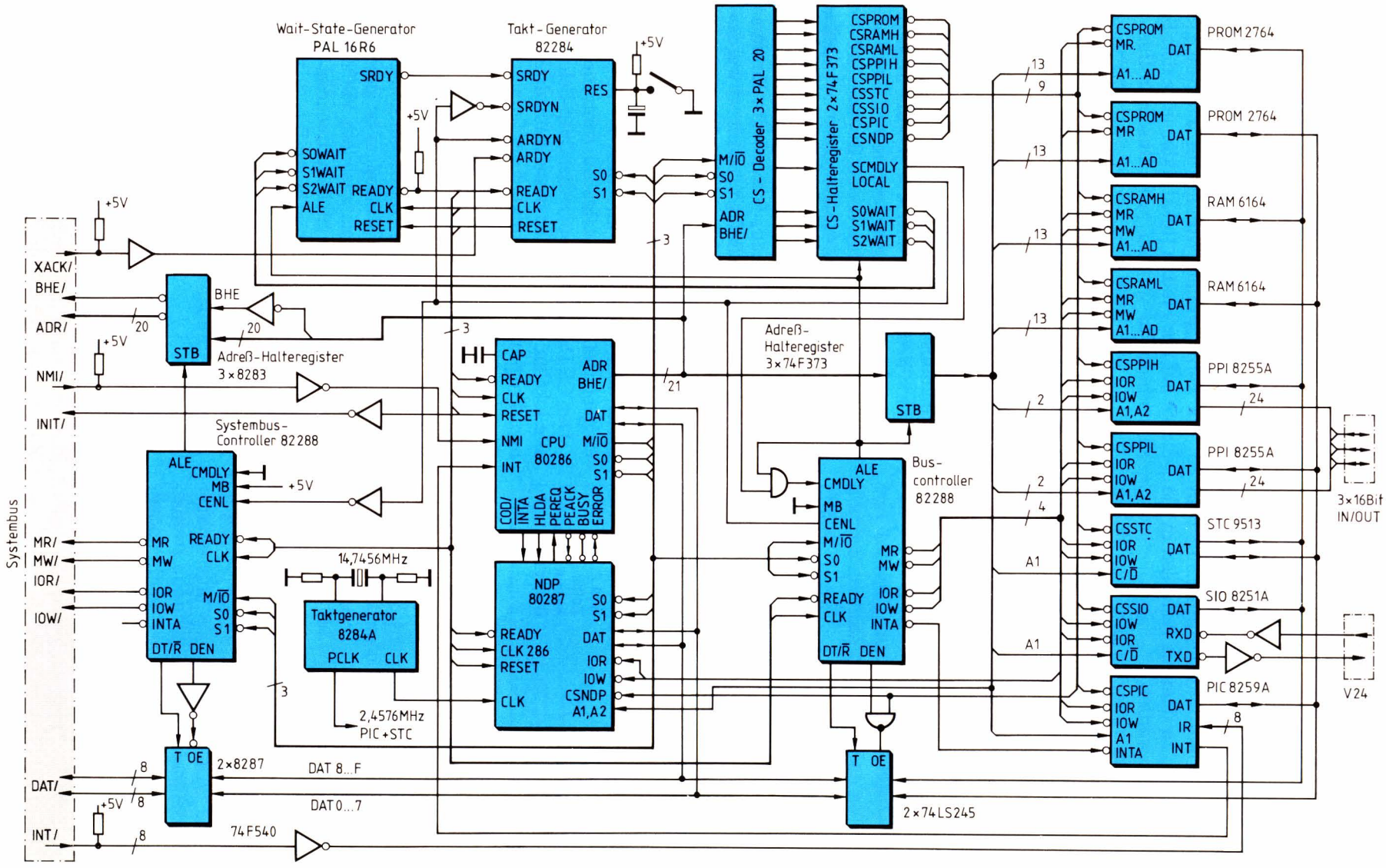


Bild 3. Blockschaltung der CPU-Platine mit einem iAPX 286

Wait-State-Generator und das „Data-Float-Time"-Problem

Der Wait-State-Generator besteht aus einem PAL 16R6, das sechs D-Flipflops enthält, die zu einem Schieberegister zusammengeschaltet sind. Zu Beginn eines jeden Buszyklus mit dem Erscheinen von ALE beginnt eine „eins" durch das Register zu laufen, die nach einer mit S0WAIT bis S2WAIT voreingestellten Anzahl von Taktperioden am Ausgang erscheint und SRDY steuert.

Wird die CPU mit 6 MHz betrieben (Quarzfrequenz = 12 MHz), so können Speicherbausteine ohne Wait-States angesprochen werden, wenn sie eine Adreß- und CS-Zugriffzeit von weniger als 200 ns haben und mit Schreib- und Leseimpulsbreiten von 150 ns auskommen. Von den beiden RAMs 6164-15 wird diese Forderung erfüllt, während für die verwendeten EPROMs 2764-3 ein Wait-State eingefügt werden muß. Dann darf die Zugriffszeit 360 ns betragen und es steht eine Leseimpulsbreite von 310 ns zur Verfügung. Ohne Verwendung des CMDLY kann es vorkommen, daß das Schreibkommando aktiv wird, bevor die Daten stabil sind. Die RAMs arbeiten trotzdem einwandfrei, solange sichergestellt ist, daß die Daten 60 ns vor dem Ausschalten des Schreibkommandos ihren endgültigen Wert angenommen haben.

Um sicher zu gehen, daß die Daten bei Beginn eines Schreibkommandos stabil sind, wird für alle Ein-/Ausgabebausteine ein CMDLY eingefügt, wodurch dann aber 2 Wait-States notwendig werden. Auf diese Weise können Adreß- und CS-Zugriffszeiten von 520 ns zugelassen werden und es stehen Schreib- und Lesekommandos von 400 ns Breite zur Verfügung.

Bei der Berechnung der Zeiten stößt man auf ein Problem, das durch das Einfügen von CMDLY und Wait-States nicht zu lösen ist. Viele Bausteine gehen mit ihren Datenausgängen nach einer Leseoperation zu langsam in den hochohmigen Zustand. Der PPI 8255A-5 und der SIO 8251A benötigen dazu z. B. 100 ns, während der Buscontroller im ungünstigsten Fall bereits nach 66 ns die Richtung des Datenbustreibers umschaltet und dieser die im nächsten Zyklus zu schreibenden Daten liefert. Dieser Fall tritt dann ein, wenn Lese- und Schreibzyklus auf der gleichen Platine unmittelbar aufeinanderfolgen. Er kann durch eine Zusatzlogik verhindert werden, die das Einschalten des Datenbustreibers bei Schreiboperationen verzögert.

Bei der Entwicklung der vorliegenden Platine wurde davon ausgegangen, daß dieser Fall nur bei MOVE-String-Befehlen auftritt. Nur in diesen Fällen schaltet die CPU unmittelbar zwischen TC- und TS-Phasen um, ohne TI-Phasen einzufügen. Deshalb können MOVE-String-Befehle auf dieser Platine nur im RAM ausgeführt werden (der Bus wird in 50 ns freigegeben), aber nicht zwischen PROM und RAM, weil das PROM 2764-3 105 ns dazu braucht.

Arithmetikprozessor

Der Arithmetikprozessor 80287 wird von einem eigenen Taktgenerator 8284 mit ca. 5 MHz versorgt, weil es z. Zt. keine für 6 MHz geeignete Ausführung gibt, die mit dem Takt der CPU arbeiten könnte. Der 80287 hat im Gegensatz zu seinem Vorgänger 8087 keine Möglichkeit auf die Operanden im Speicher selbst zuzugreifen, sondern er wird von der CPU über drei reservierte Adressen im E/A-Bereich (FA, FC und FE) mit Daten und Befehlen versorgt. Das ist deshalb notwendig, weil in der Betriebsart mit Speicherschutz nur die CPU Kenntnis über die Bereiche hat, auf die momentan zugegriffen werden darf.

Die Übertragbarkeit der Software

Obwohl der iAPX 286 die Programme des iAPX 86 verarbeiten kann, ist einige Vorsicht geboten. Wegen der höheren Geschwindigkeit des iAPX 286 kann es z. B. bei der Initialisierung von Ein-/Ausgabebausteinen vorkommen, daß diese unmittelbar aufeinanderfolgende Steuerkommandos nicht sicher erkennen, weil der zulässige minimale Abstand zwischen zwei Schreiboperationen unterschritten wurde. Enthalten die Programme Abschnitte, bei denen es auf bestimmte Ausführungszeiten ankommt, so sind ganz sicher Schwierigkeiten zu erwarten.

Wird bei einer DIV-Instruktion der zulässige Bereich für das Ergebnis überschritten, so wird ein Interrupt über den Vektor #0 ausgelöst. Im iAPX 86 enthält der Stack beim Erreichen der Interruptprozedur einen Zeiger auf die Instruktion nach der DIV-Instruktion, die den Fehler verursachte. Der iAPX 286 dagegen liefert einen Zeiger auf die fehlerauslösende DIV-Instruktion, wodurch die Interruptprozedur gezwungen wird, die Operanden für die DIV-Instruktion so zu verändern, daß sie fehlerfrei durchlaufen werden kann. Andernfalls wird die Interruptprozedur immer wieder angesprungen.

Es ist ratsam, zusätzliche Interruptprozeduren mit den dazugehörenden Vektoren vorzusehen, die beim Auftreten von nichtdefiniertem Maschinencode (#6) und bei Fehlermeldungen des 80287 (#16) aufgerufen werden. Eine ausführliche Beschreibung weiterer Unterschiede ist in (3) enthalten.

Jürgen Petsch

Literatur

[1] Geyer, J.: Schneller Prozessor arbeitet mit virtuellem Adreßbereich. ELEKTRONIK 1983, H. 5, S. 59...68.

[2] iAPX 286 Hardware Reference Manual, Handbuch der Firma Intel.

[3] iAPX 286 Programmer's Reference Manual, Handbuch der Firma Intel.

Serielle Ausgabe-Erweiterung für Einchip-Mikrocomputer

Einchip-Mikrocomputer stellen auch für die Konsumelektronik kostengünstige Lösungen dar, so daß sie in diesem Bereich zunehmend eingesetzt werden (Beispiele: Waschmaschinen, Telefonwählautomaten, Rundfunk- und Fernsehgeräte). Dem Anwender steht bei Einchip-Mikrocomputern nur eine begrenzte Anzahl von Port-Leitungen zur Verfügung. Am konkreten Beispiel einer 20stelligen Anzeige wird in diesem Beitrag gezeigt, wie auf einfache Weise ohne viel Hardware-Aufwand zusätzlich zu den Port-Leitungen eines Mikrocomputers praktisch unbegrenzt viele Ausgabeleitungen gewonnen werden können.

Die parallele Ansteuerung eines Anzeigenfeldes durch den Mikrocomputer belegt eine vom Umfang des Anzeigenfeldes abhängige größere Anzahl von Port-Leitungen des Mikrocomputers. Steuert man allerdings das Anzeigenfeld seriell an, so werden nur wenige Leitungen des Mikrocomputers für die Aufgabe benötigt. Die Anzahl der Leitungen ist zudem unabhängig von der Anzahl der zu bedienenden Anzeigen, so daß der Vorteil bei der Verwendung seriell ansteuerbarer Anzeigeeinheiten mit wachsender Stellenzahl gegenüber der parallelen Lösung zunimmt. Liegen Anzeigeeinheit und Mikrocomputer räumlich voneinander getrennt, sind diese mit nur drei Leitungen miteinander zu verbinden. Abschirmungsaufwand und Kabelkosten verringern sich gegenüber einer parallelen Lösung beträchtlich.

Es soll die Anschaltung des seriell ansteuerbaren Anzeigentreiberbausteins SDA 2014 an einen Mikrocomputer beschrieben werden. Für den Mikrocomputer-Baustein SAB 8051 wird die Anschaltung sowie ein kurzes Programm angegeben, das zehn Bytes an zwanzig Sieben-Segment-Anzeigen (z. B. 20stellige Telefonnummer eines Wählcomputers) überträgt.

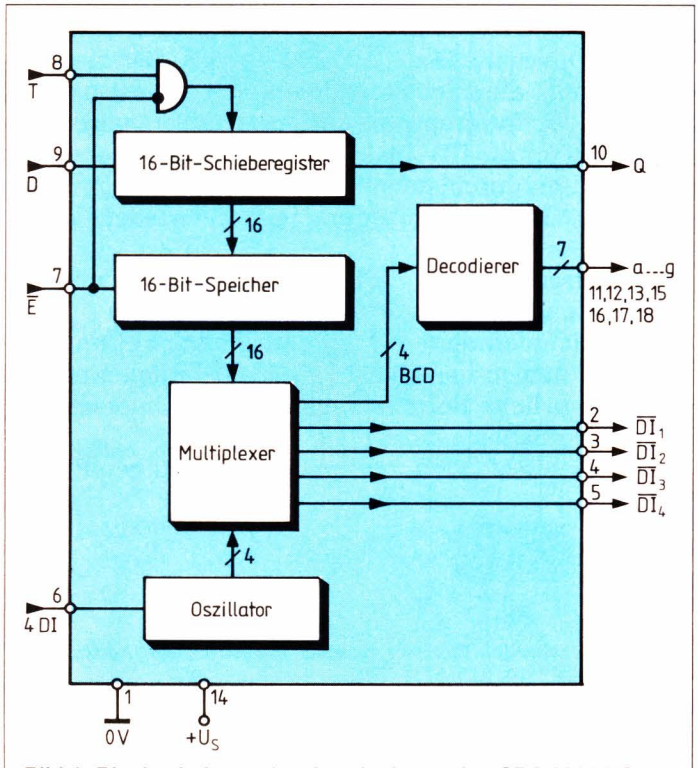


Bild 1. Blockschaltung des Anzeigebausteins SDA 2014 [3]

Anzeigebaustein SDA 2014

Im Baustein SDA 2014 sind ein serielles 16-Bit-Schieberegister, ein paralleler 16-Bit-Speicher, BCD-7-Segment-Decodierer und Treiber für Sieben-Segment-LED-Anzeigen integriert. Mit wenig Zusatzaufwand, nämlich sieben Strombegrenzungswiderständen, lassen sich zwei beziehungsweise vier Sieben-Segment-Anzeigen (Steuerung über Anschluß 4DI) betreiben. Die Anzeigen werden mit Hilfe des internen Oszillators selbständig im Multiplexbetrieb geschaltet. Ein Auszug aus dem Datenbuch [3], das Blockschaltbild des SDA 2014, ist in *Bild 1* wiedergegeben. Zur Dateneingabe von einem Mikrocomputer an den SDA 2014 benötigt man den Eingang D, an dem die Daten seriell angeboten werden müssen, den Eingang T, an dem der Begleittakt zu den Daten anliegen muß, sowie den Eingang $\overline{E}$, mit dem die Übernahme der Information aus dem Schieberegister in die Parallel-Speicher gesteuert wird. Der Ausgang Q des Schieberegisters ermöglicht die Kaskadierung mehrerer Bausteine

des Typs SDA 2014, so daß beliebig viele Anzeigen von der seriellen Datenquelle aus bedient werden können.

SDA 2014 an einem Mikrocomputer

Die Anschaltung des Bausteins SDA 2014 an einen Mikrocomputer zeigt das *Bild 2*.

Bei der Anschaltung des Bausteins sind folgende Punkte zu beachten:

- das höchstwertige Bit jeder Ziffer ist zuerst zu senden;
- der Taktausgang muß im Ruhezustand sein;
- bei der negativen Flanke am Takteingang muß die Information am Eingang D des seriellen Schieberegisters gültig sein;
- die steigende Flanke von $\overline{E}$ triggert die Übernahme der Information aus dem Schieberegister in die parallelen Speicher.

Diese Bedingungen werden durch das Impulsdiagramm des SDA 2014 [3] in *Bild 3* wiedergegeben.

Zusammenschaltung SDA 2014/SAB 8051

Als konkretes Beispiel der Zusammenschaltung eines Mikrocomputers mit dem seriellen Anzeigebaustein nach *Bild 2* soll die Anschaltung eines Anzeigefeldes an den Mikrocomputer SAB 8051 gezeigt werden.

Folgende Verbindungen sind zwischen SDA 2014 und SAB 8051 herzustellen:

Leitung 0 von Port 3 → Dateneingang D am ersten SDA 2014
Leitung 1 von Port 3 → Takt-Eingang T aller SDA 2014
Leitung 2 von Port 3 → Übernahmeeingang $\overline{E}$ aller SDA 2014

Diese Leitungen können unter den Port-Leitungen des SAB 8051 beliebig gewählt werden. Die getroffene Auswahl ist auf das Programm in *Bild 4* abgestimmt.

Zur Kaskadierung sind von den SDA 2014 die Ausgänge Q mit den Dateneingängen D der folgenden Stufen miteinander zu verbinden.

Das für den SAB 8051 geschriebene Programm von Bild 4 gibt zehn Bytes des internen Datenspeichers an ein Anzeigenfeld mit fünf SDA 2014, das entspricht einem Anzeigenfeld mit zwanzig Sieben-Segment-Anzeigen, aus.

Die integrierte serielle Schnittstelle des SAB 8051 konnte hier nicht verwendet werden, da der SAB 8051 in der Betriebsart 0 der seriellen Schnittstelle den Taktausgang im Ruhezustand auf High-Pegel legt und zudem der SAB 8051 bei voller Geschwindigkeit zu schnell für den SDA 2014 ist. Das Programm von Bild 4 wurde für das Kursmodell „Wetterstation", das einen mit 6-MHz-Quarzfrequenz betriebenen SAB 8031 (8051 ohne Programmspeicher) benutzt, an der Siemens-Schule für Mikrocomputer [5] entworfen.

Schlußbemerkungen

Das gezeigte Beispiel erweitert die Ausgabe eines Mikrocomputers um eine praktisch unbegrenzte Anzahl von Ziffernanzeigen. Einige dem Baustein SDA 2014 ähnliche Bausteine ermöglichen die serielle Ausgabe-Erweiterung auch zu beliebigen anderen Zwecken wie zum Beispiel zur Ansteuerung von Relais und Optokopplern:

SDA 2004 Anzeigebaustein für vier Sieben-Segment-LED-Anzeigen mit einer Funktion ähnlich wie der SDA 2014, jedoch einige Sonderzeichen (U, A, –).

SDA 2130 Universelle Ausgabe-Erweiterung über sechzehn Ausgangsstufen mit einstellbarem Treiberstrom.

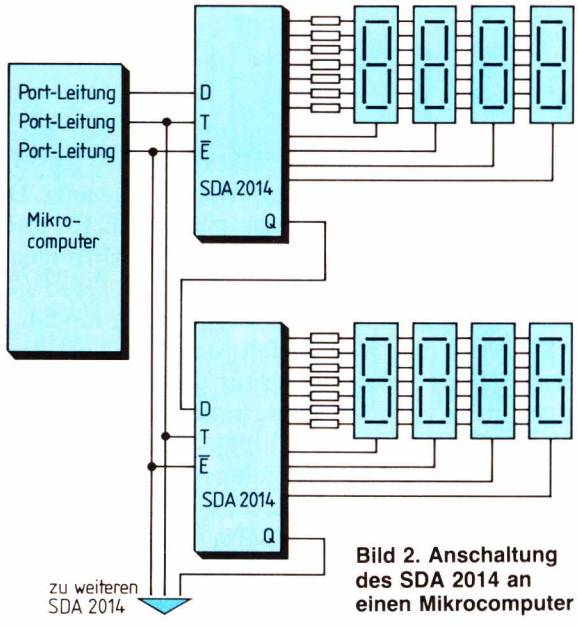


Bild 2. Anschaltung des SDA 2014 an einen Mikrocomputer

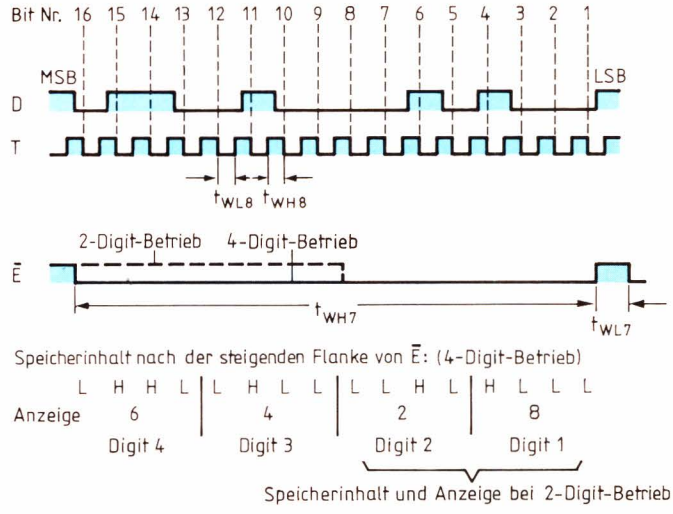


Bild 3. Impulsverlauf an den Eingängen des Anzeigebausteins SDA 2014

```
Anzeigen:     CLR   P3.2                  ; Uebernahmeleitung in Ruhezustand bringen
              CLR   P3.1                  ; Taktleitung in Ruhezustand bringen

              MOV   R4,#10                ; Anzahl der Speicherzellen
              MOV   R0,#Anzeigen_anfang   ; Anfangsadresse des Anzeigenbereichs

Neues_Byte:   MOV   R3,#08                ; Bit Zaehler
              MOV   A,@R0
              INC   R0

Neues_Bit:    RLC   A
              MOV   P3.0,C                ; Daten ausgeben

Takt_Impuls:  SETB  P3.1                  ; Takt- Impuls ausgeben
              CLR   P3.1

              DJNZ  R3,Neues_Bit
              DJNZ  R4,Neues_Byte

              SETB  P3.2                  ; Uebernahme- Impuls ausgeben
              CLR   P3.2

              RET
```

Bild 4. Unterprogramm zur Ausgabe von 10 Bytes an ein 20stelliges Anzeigenfeld

SDA 2131 Funktion ähnlich wie SDA 2130, jedoch mit festem Treiberstrom (10 mA) insbesondere für LED-Ansteuerungen geeignet.

Die Ansteuerung dieser Bausteine unterscheidet sich geringfügig von der Ansteuerung des SDA 2014, so daß das angegebene Programm leicht angepaßt werden kann.

Klaus-Peter Köhn

Literatur

[1] SAB 8051 8 Bit Einchip-Mikrocomputer (Siemens Datenblatt).
[2] SAB 8051 Family of Single Chip Mikrocomputers User's Manual.
[3] Integrierte Schaltungen für die Unterhaltungselektronik 1980/81 (Siemens).
[4] ICs für die Unterhaltungselektronik 80/81, Nachtrag z. Datenbuch 80/81.
[5] Kursunterlagen Assemblerkurs 51, Siemens-Schule für Mikrocomputer.
[6] *Köhn, K.-P., Krauss, K.-H.:* Die Bit-Verarbeitungsbefehle des SAB 8051. Markt & Technik 1982, H. 4, S. 40.
[7] *Bechen, P. v.:* Einchip-Computer-Familie für anspruchsvolle Aufgaben. ELEKTRONIK 1980, H. 22, S. 95...98.

Adreß-Tracer

Ein Adreß-Tracer speichert fortlaufend die Adressen eines Mikroprozessorsystems, bis eine vorher eingestellte Stoppadresse erreicht ist. Danach kann der Tracer-Speicher rückwärts ausgelesen werden.

Der Adreßbus A0...15 (*Bild*, nächste Seite) liegt am 16-Bit-Vergleicher IC1...4 an. Gleichzeitig ist er über Treiber (IC5...6) mit den Datenleitungen des Tracer-Speichers (IC7...10) und vier Anzeigebausteinen (IC17...20) verbunden. Da als Speicher die handelsüblichen Bausteine des Typs 2114 (1024 × 4) verwendet werden, benötigt man noch einen Zähler (IC11...13), der als Ringzähler (modulo 1024) die Adressen des Tracer-Speichers bildet. Diese zehn Adressen (C0...9) werden nun noch auf drei Anzeigebausteine (IC21...23) gelegt, um den aktuellen Speicheradressenstand beobachten zu können.

Die Vergleichsadresse wird über zwei 8polige DIL-Schalter an den Vergleicher gelegt. Nach Anschließen des Tracers an das Mikroprozessorsystem wird mit S2 das erste Flipflop in IC14 definiert zurückgesetzt. Damit wird Ausgang 5 low, und Ausgang 6 wird high; und der Systemtakt, der den Zähler hochzählt wird freigegeben (über Anschluß 4 von IC16). Das zweite Flipflop von IC14 dient lediglich zur Tastenentprellung für S4.

Mit S5 wird der Zähler definiert auf Null gesetzt. Schalter S6 dient der Umschaltung des Zählers. Mit ihm kann der Zähler vorwärts oder rückwärts betrieben werden. S1 und S3 stehen in Stellung „Write". Nun werden laufend Adressen in den Speicher geschrieben, bis die eingestellte Stoppadresse mit der auf dem Adreßbus übereinstimmt. Als Zeichen dafür bleibt der Anzeigewert auf IC21...23 stehen. Bei Gleichheit wird Anschluß 6 von IC1 high. Dieses Signal gelangt über das Gatter I von IC16 auf den Takteingang von IC14. Da der Daten-

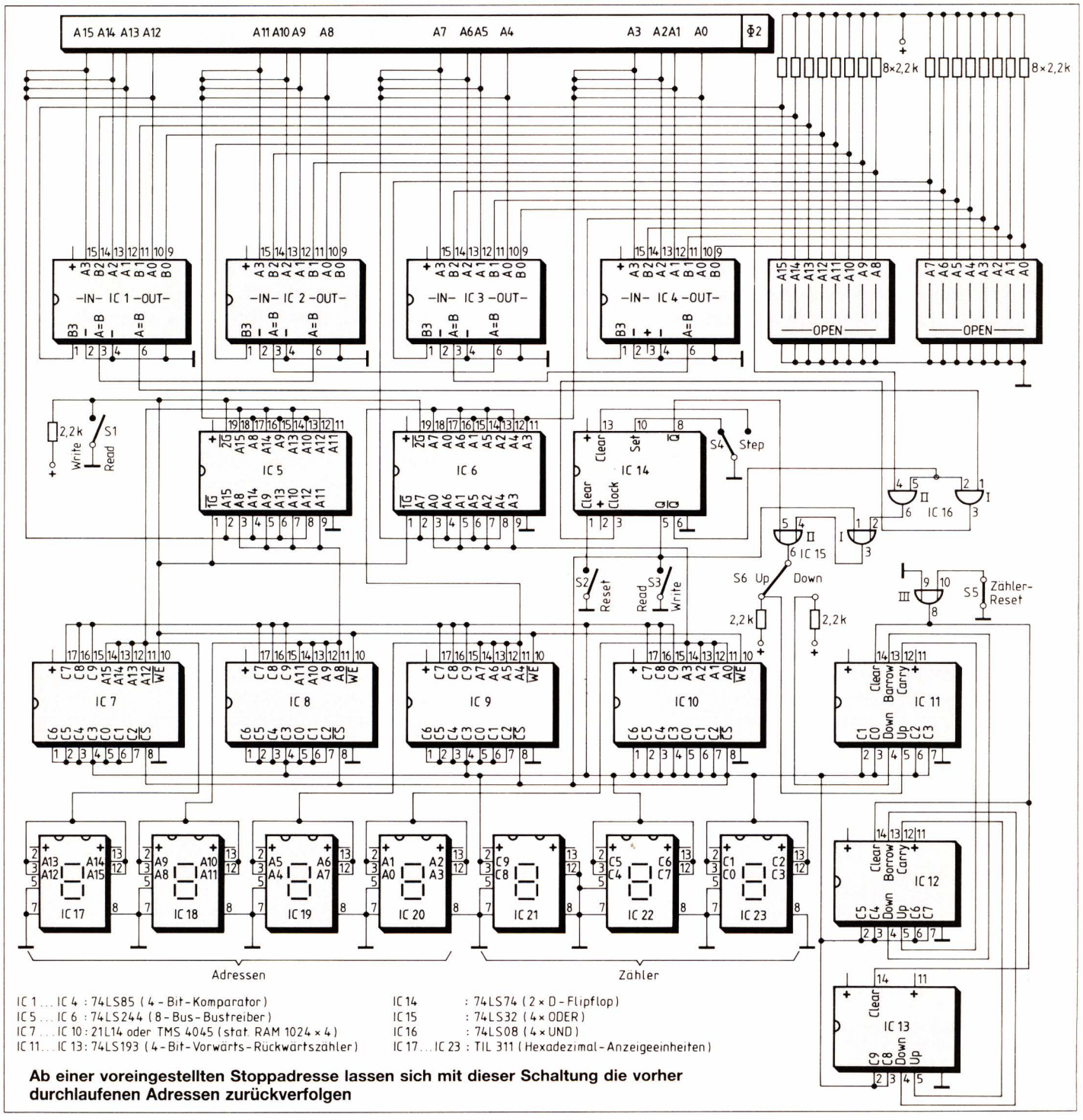
eingang von IC14 immer auf high liegt, wird Anschluß 6 low und der Systemtakt über Gatter II von IC16 gesperrt. Gleichzeitig wird Anschluß 5 von IC14 high, und er legt CS von IC7...10 auf high, die damit gesperrt sind. Nun wird S6 umgeschaltet, damit nachher die Adressen rückwärts ausgelesen werden können.

Zum Lesen des Speichers werden S1 und S3 auf „Read" geschaltet. Damit werden IC5 und IC6 hochohmig; gleichzeitig wird CS für den Speicher erzeugt. Nun kann mit S4 der Inhalt des Speichers Schritt für Schritt rückwärts ausgelesen und angezeigt werden.

Der Tracer ist in erster Linie dazu gedacht, selbsterstellte Programme auszutesten. Auch umfangreiche Betriebsroutinen können verfolgt werden, da eine Speichertiefe von 1024 Adressen im Regelfall genügen sollte.

Die Schaltung wurde für den Prozessor 6502 gebaut. Zum Hochzählen des Zählers ist nur Φ2 maßgebend. Um die Schaltung auch für andere Prozessoren einsetzen zu können, muß man ein Signal erzeugen, das nur dann high wird, wenn die Adresse auf dem Adreßbus sicher anliegt. Dies dürfte im Normalfall keinerlei Schwierigkeiten bereiten.

Hubertus Arndt



IC 1 ... IC 4 : 74LS85 (4-Bit-Komparator)
IC 5 ... IC 6 : 74LS244 (8-Bus-Bustreiber)
IC 7 ... IC 10 : 21L14 oder TMS 4045 (stat. RAM 1024 × 4)
IC 11 ... IC 13 : 74LS193 (4-Bit-Vorwärts-Rückwärtszähler)
IC 14 : 74LS74 (2 × D-Flipflop)
IC 15 : 74LS32 (4 × ODER)
IC 16 : 74LS08 (4 × UND)
IC 17 ... IC 23 : TIL 311 (Hexadezimal-Anzeigeeinheiten)

Ab einer voreingestellten Stoppadresse lassen sich mit dieser Schaltung die vorher durchlaufenen Adressen zurückverfolgen

Mikrocontroller besitzt BASIC-Interpreter

Mikrocontroller werden auch als „Computer auf einem Chip" bezeichnet. Bei den meisten Bauelementen dieser Art handelt es sich aber um Chips, die lediglich über dedizierte Programme verfügen und nicht als universelle Computer zu bezeichnen sind. Nachdem es möglich ist, auch größere Softwarepakete „in Silizium" unterzubringen, bereitet es keine Probleme, einen Controller mit einem vollständigen BASIC-Interpreter auszustatten. Beispiel dafür ist das Mikrocontroller-System MCS BASIC-52 von Intel.

Im Gegensatz zu abgemagerten BASIC-Interpretern, die bereits auf dem Markt erhältlich sind, ist MCS BASIC-52 ein vollständiger Interpreter, der Merkmale wie z. B. Fließkommaarithmetik, EPROM-Programmierroutinen, logische Operatoren (UND, ODER, EXOR und Inversion), die Möglichkeit, Zahlen in Fließkommaformat oder Hexadezimal-Format auszugeben, mnemonischer Zugriff auf alle E/A-Ressourcen, Zeichenketten-Manipulation und Echtzeit-Taktgenerator mit Auflösung von 5 ms sowie eine vollständige Library von Routinen, auf die über Assemblersprache zugegriffen werden kann, bietet. *Tabelle 1* zeigt den Befehlssatz. Neben den standardmäßigen Befehlen verfügt MCS BASIC-52 über zusätzliche Instruktionen, die insbesondere für den Betrieb in eingebettetem System geeignet sind.

Das Software-Paket MCS BASIC-52 befindet sich im ROM-Bereich von 8 KByte auf dem Controller-Chip 8052AH, der zur MCS-51-Familie gehört. Dieser Baustein enthält alle Merkmale des 8051, darüber hinaus aber noch ein ROM von 4 KByte, 128 Byte RAM sowie einen 16-Bit-Universalzähler/Zeitgeber.

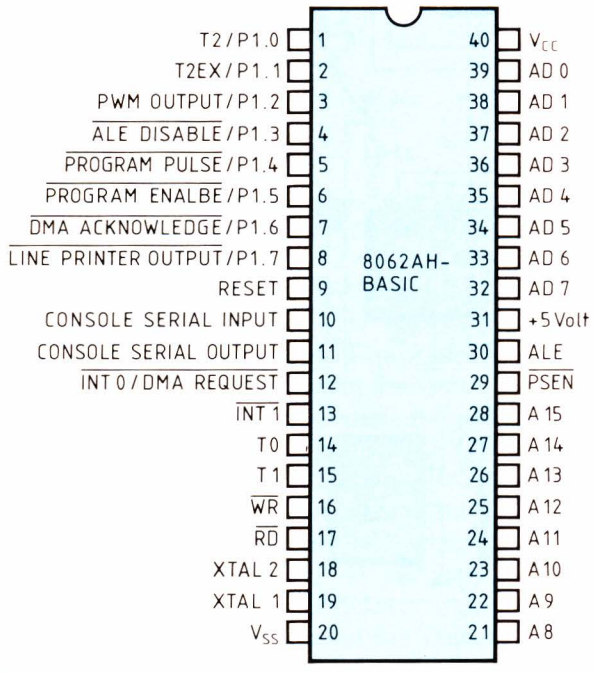


Bild 1. Anschlußbelegung des Bausteins 8052AH. Dieser Chip enthält das Softwarepaket MCS BASIC-52. Wenn Anschluß 31 auf Masse liegt, arbeitet dieser Baustein wie der Standard-Chip 8032AH

Warum BASIC?

BASIC hat, wie alle anderen Computersprachen, gewisse Vorzüge und Nachteile. Besonders interessant ist BASIC allerdings, weil es sich um eine Programmiersprache handelt, die für viele Anwendungsbereiche zum Standard geworden ist. Die weite Verbreitung hat ihren Grund in ihrem einfachen Aufbau und ihrer leichten Programmierbarkeit. Die offensichtlichen Nachteile und Begrenzungen werden bei BASIC-52 durch bestimmte Befehle ausgeglichen.

Aufbau des Controllers, wichtigste Merkmale

Wie bereits erwähnt, befindet sich das Software-Paket MCS BASIC-52 in einem standardmäßigen 40poligen DIL-Gehäuse (*Bild 1*). Das BASIC hat einige der Anschlüsse für bestimmte Funktionen fest belegt. Die Stifte 1...8 werden als E/A-Anschlüsse für den 8052AH benutzt und sind mit „PORT 1" (P1) bezeichnet. Über

die Stifte 1 und 2 läßt sich auch der Zeitgeber/Zähler 2 takten und ansteuern. MCS BASIC-52 kann auch die Anschlüsse des PORT 1 als Standard-E/A-Verbindung benutzen, wobei die Syntax des verwendeten BASIC direkte Lese- und Schreib-Vorgänge für PORT 1 erlaubt. Beispielsweise bedeutet das Statement:

PORT 1 = 55 H,

daß der Wert 55 H auf den Anschlüssen des PORT 1 anliegt und das Statement:

A = PORT 1

bedeutet, daß die Variable A den Anschlüssen des PORT 1 zugeordnet wird. Außer als E/A-Port kann man diese Anschlüsse auch mit Hilfe bestimmter Kommandos und Statements für weitere nützliche Zwecke verwenden. So benutzt man beispielsweise Bit 2 von Port 1 (P1.2) als Ausgangsanschluß für das PWM-Statement (Puls-Weiten-Modulation). Mit diesem Befehl kann man eine Impulsgruppe mit veränderbarer Frequenz, Dauer und variablem Tastverhältnis erzeugen. Zum Beispiel ergibt sich mit dem Befehl:

PWM 5+, 100, 1000

ein Impulspaket von 1000 Zyklen, das für 50 µs (bei 12-MHz-Takt) auf High-Pegel liegt und für 100 µs auf Low-Pegel. Das PWM-Statement eignet sich insbesondere für die Erzeugung von Tonsignalen in Sicherheitssystemen.

Die Bits 3, 4 und 5 von Port 1 (P1.3, P1.4, P1.5) erzeugen die gesamten Zeitsignale und sorgen dafür, daß praktisch jedes EPROM- und E^2PROM-Bauelement zu programmieren ist. Hiermit ergibt sich eine einfache Möglichkeit, Programme sicher abzulegen.

Bit 6 von Port 1 (P1.6) ergibt in Zusammenhang mit dem „INTO-Anschluß" (Stift 12) eine Option zur Implementierung der DMA/DAM-ACK-Funktion. Dieses Merkmal erlaubt die Benutzung eines gemeinsamen Speicherbereiches. Bit 7 von Port 1 (P1.7) wird als serieller Ausgang für eine Peripherieeinheit, z. B. für einen Drucker, benutzt. Das Kommando LIST und das Statement PRINT leiten die auszugebende Information direkt an das Peripheriegerät. Mit Hilfe des Reset-Anschlusses (Stift 9) läßt sich das System initialisieren. Ein externer Widerstand von 8,2 kΩ sorgt für automatischen Einschalt-Reset, wenn ein Kondensator von 10 µF an diesem Anschluß gegen U_{CC} liegt.

Port 3 (Anschlüsse 10...17) wird von MCS BASIC-52 nicht berücksichtigt. Die Anschlüsse werden beim 8052AH für andere Zwecke benutzt. Ein interessantes Merkmal des MCS BASIC-52 ist, daß Interrupts auf dem Anschluß INT1 (Stift 13) vom BASIC-Programm bearbeitet werden können. Das Statement ONEXT1 ermöglicht dies. Nach dem Statement

10 ONEXT 1 1000

wird z. B. jedesmal, wenn der Anschluß INT1 auf Logisch 0 gezogen wird, ein Unterprogramm auf Zeilennummer 1000 initiiert, mit dessen Hilfe man einen Interrupt bearbeitet. Das Statement RETI dient, aus der Interrupt-Verarbeitung zurückzukehren.

Die übrigen Anschlüsse des Bauelementes entsprechen den Funktionen des 8051. Port 0 und Port 2 lassen sich nur als Adreß-/Datenanschlüsse verwenden und nicht als E/A-Vebindung. Der Anschluß „EA" (Stift 31) muß mit U_{CC} verbunden sein, weil der Interpreter sich im ROM auf dem Chip befindet.

Tabelle 1. Befehlssatz des MCS BASIC-52

Kommandos	Statements	Operatoren
RUN	BAUD	ADD (+)
LIST	CALL	DIVIDE (/)
LIST#	CLEAR	EXPONENTIATION (**)
NEW	CLEARS	MULTIPLY (*)
NULL	CLEARI	SUBTRACT (–)
RAM	CLOCK0	LOGICAL AND (.AND.)
ROM	CLOCK1	LOGICAL OR (.OR.)
XFER	DATA	LOGICAL X-OR (.XOR.)
PROG	READ	LOGICAL NOT
PROG1	RESTORE	ABS ( )
PROG2	DIM	INT ( )
FPROG	DO-WHILE	SGN ( )
FPROG1	DO-UNTIL	SQR ( )
FPROG2	END	RND
	FOR-TO-STEP	LOG ( )
	NEXT	EXP ( )
	GOSUB	SIN ( )
	RETURN	COS ( )
	GOTO	TAN ( )
	ON-GOTO	ATN ( )
	ON-GOSUB	=, >, > =, <, < =, <>
	IF-THEN-ELSE	ASC ( )
	INPUT	CHR ( )
	LET	CBY ( )
	ONERR	DBY ( )
	ONEXT1	XBY ( )
	ONTIME	GET
	PRINT	IE
	PRINT#	IP
	PH0.	PORT1
	PH0.#	PCON
	PH1.	RCAP2
	PH1.#	T2CON
	PUSH	TCON
	POP	TMOD
	PWM	TIME
	REM	TIMER0
	RET1	TIMER1
	STOP	TIMER2
	STRING	TIME
	UI0	XTAL
	UI1	MTOP
	UO0	LEN
	UO1	FREE
		PI

Vollständiges System auf dem Chip

Weil MCS BASIC-52 ein vollständiges System auf einem Chip ist, müssen lediglich ein externes RAM sowie serielle Port-Treiber hinzugefügt werden, um ein arbeitsfähiges System aufzubauen (*Bild 2*). Das RAM muß wenigstens 1 KByte Kapazität umfassen, und die externen Speicheradressen müssen bei der Adresse 0 anfangen und kontinuierlich decodiert werden können. Nach dem Reset initialisiert MCS BASIC-52 den Spei-

cher und wartet dann darauf, daß der Benutzer ein Leerzeichen (20H) eintastet. Hiermit wird die Übertragungsfrequenz eingestellt, Schalter müssen nicht betätigt werden.

Obwohl das in Bild 2 dargestellte System brauchbar ist, handelt es sich dabei natürlich noch nicht um einen eingebetteten Controller. Mit nur wenig zusätzlicher Hardware ergibt sich allerdings eine Konfiguration, die über eine Auto-Start-Einrichtung, EPROM/E²PROM-Programmierung und einfache Dateiverwaltung verfügt (*Bild 3*).

Die Programmierung von EPROMs und E²PROMs ist eine der leistungsfähigsten Merkmale des MCS BASIC-52, insbesondere wegen der Unkompliziertheit. Lediglich drei Steueranschlüsse dienen dieser Funktion. Nachdem der Benutzer ein Programm eingegeben hat, wird dieses beginnend mit der Adresse 512 im RAM gespeichert. Wenn der Benutzer „PROG" oder „FPROG" eintippt, programmiert das System den RAM-Inhalt in ein EPROM/E²PROM, das unter 8000H adressiert wird. Um den Programmierprozeß zu starten, schreibt die CPU zunächst eine Logische 0 in den „Program-Enable"-Anschluß (P1.5). Mit seiner Hilfe wird die höhere Programmierspannung freigegeben. Zusätzlich läßt sich dieses Signal dazu benutzen, U_{CC} auf 6 V zu erhöhen, was für den neuen „Intelligent"-Programmieralgorithmus von Intel erforderlich ist. Danach liest der Prozessor die Informationen aus den RAM-Plätzen; die entsprechenden EPROM/E²PROM-Adressen werden auf den Anschlüssen AD0...AD7 des 8052AH ausgegeben. ALE-Disable wird auf 0 gesetzt. Dadurch werden die unteren EPROM/E²PROM-Adressen im System-Adreß-Latch zwischengespeichert (74LS373). Für diese Funktion ist lediglich ein UND-Gatter erforderlich. Der Prozessor schreibt die höheren EPROM/E²PROM-Adressen auf die Anschlüsse A8...A15 und die Daten auf die Anschlüsse AD0...AD7. Zum Schluß wird eine Logische 0 auf den „Program-Pulse"-Anschluß gegeben. Je nach verwendetem Algorithmus liegt dieser Anschluß entweder für 1 oder 50 ms auf Low. Nachdem dort wieder der Logikpegel für 1 anliegt, verifiziert die CPU den Inhalt des programmierten EPROMs und schreibt daraufhin eine 1 auf den ALE-Disable-Stift.

Der Programmiervorgang läuft, bis das gemeldete BASIC-Programm im EPROM/E²PROM abgelegt ist. Danach schreibt die CPU eine 1 auf den Program-Enable-Anschluß und verläßt die Programmierroutine, um zur „BASIC-Command"-Betriebsart zurückzukehren. Es können mehrere Programmieralgorithmen realisiert werden. Die meisten Standard-EPROMs lassen sich mit dem 50-µs-Impuls programmieren. Eine Impulsdauer von 1 µs ist für die neue Generation EPROMs hoher Schaltungsdichte mit dem Intelligent-Algorithmus erforderlich. Um mit diesem schnellen Algorithmus

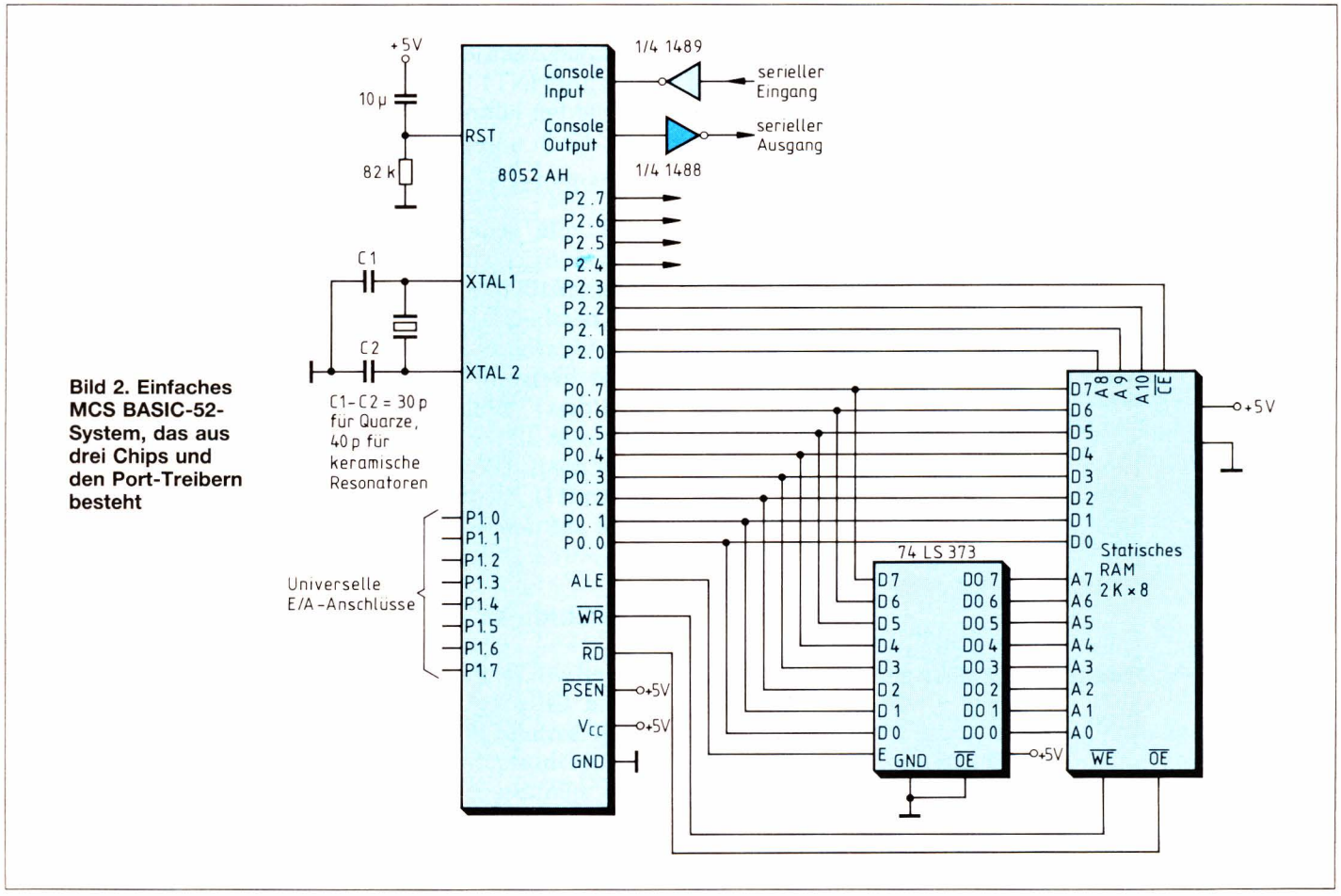


Bild 2. Einfaches MCS BASIC-52-System, das aus drei Chips und den Port-Treibern besteht

arbeiten zu können, gibt der Benutzer anstelle von PROG das FPROG ein. MCS BASIC-52 erlaubt es darüber hinaus, jede beliebige Länge für EPROM/E²PROM-Programmierimpulse generieren zu können.

Um unabhängig von der Genauigkeit des Systemtaktes zu sein, berechnet MCS BASIC-52 alle kritischen Zeitparameter aus der Variablen „XTAL", die der Benutzer angibt. Nach dem Reset setzt MCS BASIC-52 einen Systemtakt von 11,0592 MHz voraus. Wenn ein Benutzer folgendes eintippt:

XTAL = 12000000,

nimmt MCS BASIC-52 einen Systemtakt von 12 MHz an und berechnet daraus EPROM/E²PROM-Programmierzeit, außerdem die Echtzeit-Taktfrequenz sowie die Baudrate für das Zeilendrucker-Port. Diese Einrichtung sorgt dafür, daß ein Entwickler sich nicht um die internen Zeitabläufe des MCS BASIC-52 zu kümmern braucht.

Ein weiteres nützliches Merkmal der EPROM/E²PROM-Programmierung ist die Tatsache, daß mehr als ein Programm in einem EPROM/E²PROM unterzubringen ist. In der Praxis lassen sich bis zu 255 Programme im System speichern, wobei allerdings die üblichen Speicherkapazitäten die Zahl der Programme begrenzt (z. B. 32 K bei EPROM/E²PROMs). Immer, wenn PROG (oder FPROG) eingetippt wird, antwortet MCS BASIC-52 mit einer Nummer, die der Datei-Nummer entspricht, die von MCS BASIC-52 dem Programm zugeordnet wurde. Um ein in der Datei befindliches Programm auszuführen, gibt der Benutzer lediglich „ROM X" ein, wobei X eine ganzzahlige Dateinummer ist. MCS BASIC-52 findet dann das entsprechende Programm. Der Benutzer muß lediglich „RUN" eintippen, das Programm beginnt dann mit der Ausführung.

Im Falle, daß ein Fehler im Programm entdeckt wird, das sich bereits im EPROM befindet, kann dieses Programm mit Hilfe des Befehls „XFER" ins RAM transferiert werden. Nach der Korrektur läßt sich das entsprechende Programm sofort in das EPROM zurückladen.

MCS BASIC-52 verfügt über weitere Programmierfunktionen für EPROMs und E²PROMs. Wie bereits erwähnt, wartet der Controller nach einem Reset auf ein Leerzeichen. Als Option kann auch die serielle Baudraten-Information in einem EPROM oder E²PROM enthalten sein. Dies erreicht man durch Eintasten von PROG1

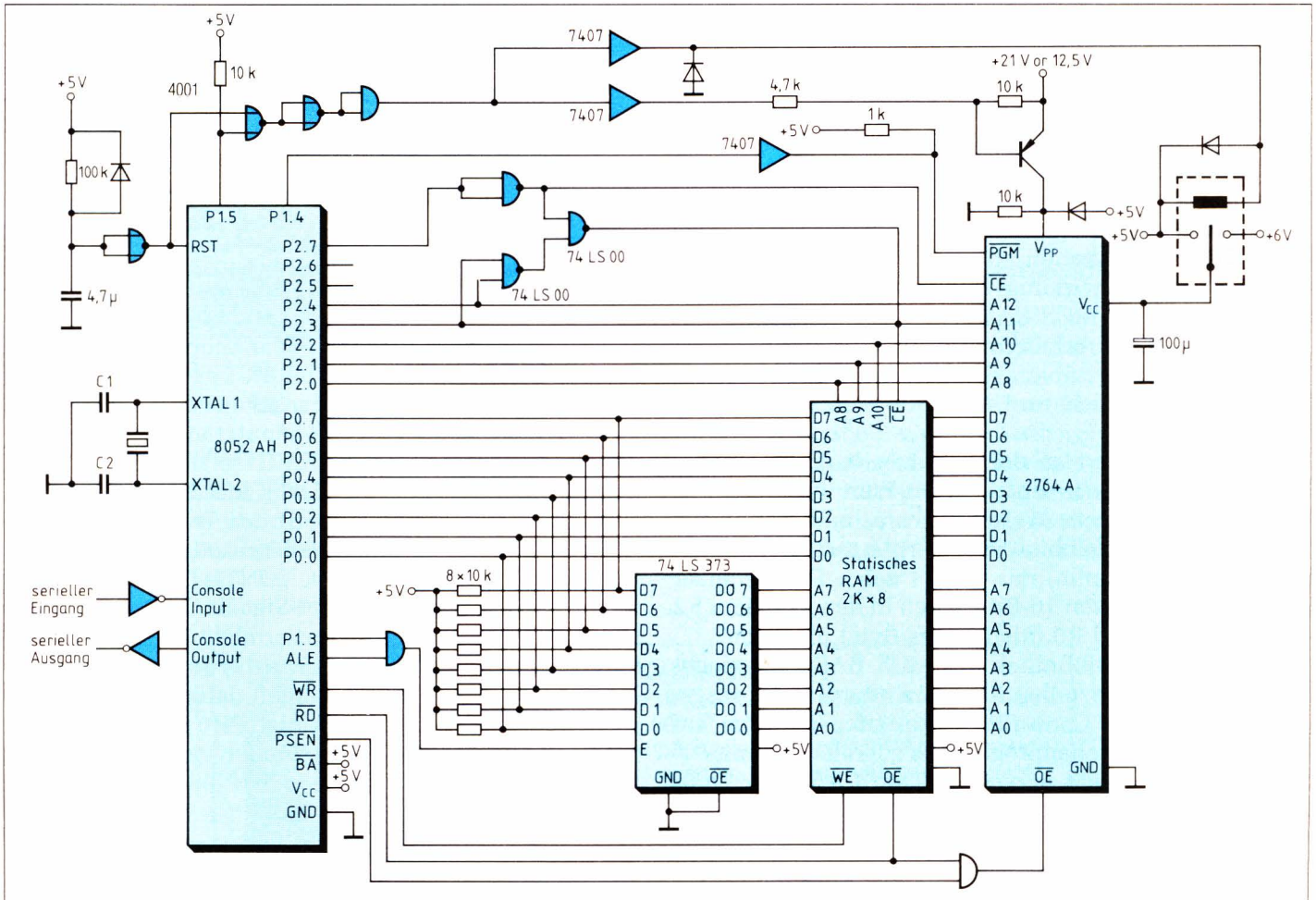


Bild 3. Vollständiges MCS BASIC-52-System mit EPROM-Programmiereinrichtung. Von dieser Konfiguration wird auch der intelligente Programmier-Algorithmus unterstützt, der die Programmierzeit größerer Speicherbausteine merklich verkürzt. Der CMOS-Baustein 4001 verhindert unerwünschtes Programmieren während des Einschaltvorganges

oder FPROG1, wenn der nächste Reset auftritt, muß kein Leerzeichen eingetastet werden.

Das wohl leistungsfähigste Merkmal des MCS BASIC-52 ist die Auto-Start-Funktion. Diese wird durch Eintasten von PROG2 oder FPROG2 in die Tastatur aufgerufen. Auto-Start legt die Baudraten-Information in der gleichen Weise ab wie es bei PROG1 oder FPROG1 erfolgt. Darüber hinaus sorgt dieser Befehl dafür, daß das erste Programm, das sich im EPROM/E²PROM befindet, direkt nach dem Reset ausgeführt wird.

Damit ist auch ein Eintippen des Kommandos „RUN" nicht mehr erforderlich. Auf diese Weise wird auch eine Eingabeeinrichtung überflüssig. Dies ist besonders wichtig für eingebettete Konzepte. Programme lassen sich schreiben und fehlerfrei machen, indem ein Terminal zu Hilfe genommen wird. Wenn der Programmierer mit seinen Lösungen zufrieden ist, läßt sich die Hardware in das Gesamtsystem integrieren, und das Terminal ist nicht mehr erforderlich.

Viele weitere Merkmale

Neben EPROM/E²PROM-Programmierung, externe Interruptverarbeitung und andere bereits erwähnte Merkmale besitzt MCS BASIC-52 Funktionen, die insbesondere für eingebettete Systeme wichtig sind. So erlaubt z. B. das Statement „CALL", daß auf Assemblersprachen-Routinen von MCS BASIC-52 aus zugegriffen werden kann. In manchen Konfigurationen sind diese Routinen erforderlich, um gewisse Zeiten einhalten zu können. Wichtig bei der Verwendung von MCS BASIC-52 in Zusammenhang mit Assemblersprachen ist, daß das Assemblerprogramm alle Vorteile einer vollständigen Library von wirkungsvollen Routinen ausnutzen kann, die sich im MCS BASIC-52-Softwarepaket befinden. Der Benutzer erhält Zugriff auf diese Bibliothek, indem er einfach einen speziellen Op-Code in den Akkumulator plaziert und dann die Adresse 30H aufruft.

Wenn beispielsweise der Benutzer in einer Assemblersprachen-Routine einen 16-Bit-Wert liest und die Quadratwurzel dieses Wertes zu berechnen hat, wobei das Resultat im gleichen Speicherplatz zu stehen hat, genügt das Programm, das *Bild 4* zeigt. Der Benutzer muß dafür die ganzen 16-Bit-Zahlen in den Registern R2 (höheres Byte) und R0 (niedrigeres Byte) plazieren.

Die Funktions-Bibliothek des MCS BASIC-52 macht das Schreiben spezieller Routinen überflüssig. Insgesamt enthält dieser Controller 60 Funktionsroutinen, auf die Assemblersprachenprogramme zugreifen können.

Aus Tabelle 1 geht hervor, daß MCS BASIC-52 vollständige Steuermöglichkeiten über alle E/A- und Speicher-Ressourcen hat, die dem System zur Verfügung stehen. Um die Sache zu vereinfachen, sind diese Operationen symmetrisch. Z. B. weist der Befehl

A = DBY (0EFEH)

den Wert, der im internen Speicher des 8052AH unter der Adresse OFEH abgelegt ist, der Variablen A zu, und

DBY (0EFH) = 22 H

plaziert 22 H im internen Speicher unter der Adresse 0FEH. Das gleiche Beispiel für Symmetrie gilt für die Interrupt-Register IE und IP sowie für Zeitgeber und Zeitgeber-Konfigurierungsregister.

Einige andere spezielle Befehle des MCS BASIC-52 sind BAUD, CLEARS, CLEARI, CLOCK1, CLOCK0, ONTIME, PH0 und PH1. BAUD stellt die Baudrate für das Zeilendrucker-Port ein. Wenn man beispielsweise

```
MOV   A,#9AH    ;This instruction loads the accumulator with
                ;the opcode that will place the
                ;register pair R2:R0 onto MCS BASIC-52S
                ;arithmetic or argument stack
;
CALL  30H       ;This causes the operation to be performed
;
MOV   A,#1FH    ;This instruction specifies that the square
                ;root of the number on the stack is to be
                ;calculated, the result is returned to
                ;the stack
;
CALL  30H       ;This causes the  square root calculation
                ;to be performed
;
MOV   A,#1H     ;This opcode specifies that the number on
                ;the stack is to be truncated and converted to
                ;a 16-bit integer, the result is placed in
                ;register pair R3 (high byte) and R1 (low byte)
;
CALL  30H       ;this causes the operation to be performed.
;
                ;At this point the square root of the number
                ;that was originally in R2:R0 is in R3:R1
```

Bild 4. Der Zugriff auf die Funktions-Bibliothek des MCS BASIC-52 erfolgt über den Op-Code im Akkumulator und Aufruf des Speicherplatzes 30 H. Die hier abgebildete Routine bildet die Wurzel aus einer 16-Bit-Zahl in den Registern R2 (höheres Byte) und R0 (niedrigeres Byte). Das Resultat befindet sich in R3 (höheres Byte) und R1 (niedrigeres Byte)

die Ausgabe mit 1200 Baud machen möchte, gibt man „BAUD 1200" ein. CLELARS benutzt man zum Zurücksetzen der Stacks und CLEARI setzt alle Interrupts zurück außer dem Echtzeit-Takt. CLOCK1 gibt den Echtzeittakt frei und CLOCK0 sperrt den Takt. Der Wert des Echtzeit-Taktsignals läßt sich lesen oder zuordnen mit Hilfe des Operators „TIME". „ONTIME" benutzt man zum Erzeugen eines Interrupt-Signals, wenn der Time-Operator einen gewissen Wert erreicht oder überschreitet. ONTIME arbeitet ähnlich wie das Statement ONEX1. Die Befehle PH0 und PH1 sorgen dafür, daß eine Ausgabe in Hexadezimal-Form erfolgt. PH0 unterdrückt führende Nullen, während PH1 immer vier hexadezimale Stellen ausgibt.

So bringt man den Chip zum Laufen

Die Konstruktion eines funktionsfähigen Systems auf der Basis des MCS BASIC-52 ist ein Kinderspiel. Der Anschluß an praktisch jedes Standard-Peripheriechip

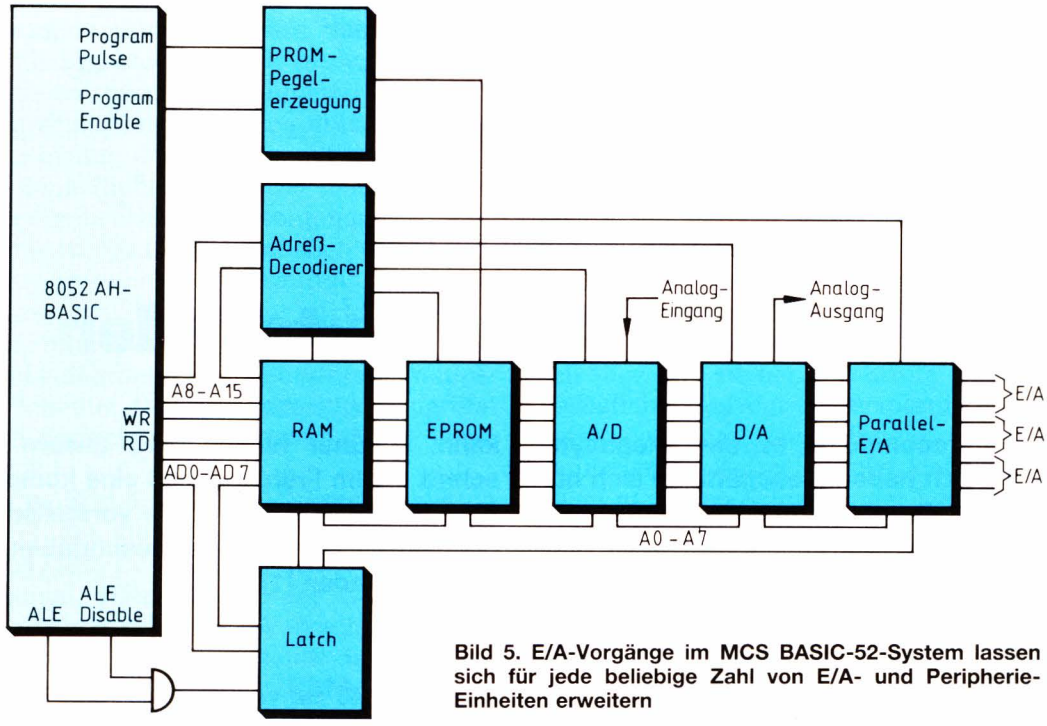


Bild 5. E/A-Vorgänge im MCS BASIC-52-System lassen sich für jede beliebige Zahl von E/A- und Peripherie-Einheiten erweitern

erfolgt durch Decodieren der Peripherieadressen, die irgendwo in den 64 KByte des externen Adreßbereiches vom 8052 liegen können. In der Praxis sind die oberen 8 KByte (56...64 KByte) des Speichers für E/A-Aufgaben reserviert. Die Zuweisung von externen Peripherieeinheiten übernimmt der XBY-(Adreß-)Operator. Beispielsweise liest

A = XBY (0FFF0H)

eine Peripherie- oder externe Speicherplatzadresse, die für die Adresse 0FFF0H decodiert ist und weist diesen Wert der Variablen A zu, während

XBY (0FFF0H) = 55 H

den Wert 55 H in eine Peripherieeinheit oder in einen Speicherplatz schreibt, der unter der Adresse 0FFF0H definiert ist. *Bild 5* zeigt die Blockschaltung eines A/D- und D/A-Umsetzers sowie eines Parallel-Ports 8255, das mit dem MCS BASIC-52 verbunden ist. Ein solches System ist typisch für diese Konfigurationen, die für die Prozeßsteuerung, Umweltüberwachung oder Energieverwaltung konzipiert sind.

Viele Möglichkeiten werden eröffnet

Die flexible und leistungsfähige Architektur des Mikrocontrollers 8052AH sorgte dafür, daß sich viele wünschenswerte und bisher einzigartige Merkmale in die Software MCS BASIC-52 unterbringen ließen. Sie alle zu beschreiben, würde allerdings den Rahmen dieses Beitrages sprengen. Ein Entwickler kann aber leicht einschätzen, wie leistungsfähig und einfach benutzbar dieses Werkzeug sein kann. Viele neue Projekte lassen sich ohne großen Aufwand verwirklichen.

John Katauski

Das 68000-Sonderheft

Dieses Sonderheft gibt einen Einblick in die Bauelementefamilie 68000, die aus verschiedenen CPU-Typen und Peripherieeinheiten besteht. Anhand praktischer Beispiele werden Konzept, Architektur, Befehlssatz und Funktion erläutert. Auch der Einsteiger in die 16/32-Bit-Technik findet wertvolle Hinweise für die Arbeit mit diesen ICs.

Das Sonderheft besteht aus bisher unveröffentlichten Beiträgen sowie überarbeiteten ELEKTRONIK-Aufsätzen. Es wendet sich an Entwickler, Programmierer und Systemintegratoren, die sich für die 68000-Familie interessieren.

Dynamische RAMs ohne Wartezyklen

Nachdem sich Mikrocomputer in den Megabyte-Bereich eingeschlichen haben, beschäftigen sich nun viele Entwickler mit Massenspeichern, die schnelle Zugriffszeiten aufweisen. Hier können Sie lesen, welche Probleme dabei auftauchen und wie man sie lösen kann. Darüber hinaus wird unsere Schwesterzeitschrift mc im Frühjahr 1986 eine komplette Schaltung einer 2-MByte-DRAM-Karte vorstellen, in der Sie die hier auftauchenden und beschriebenen Details verwirklicht finden [1].

Massenspeicher an modernen Mikrocomputersystemen – wie z. B. 68000-orientierte Rechner – stellen Hardware-Entwickler vor gänzlich neue Probleme: Im Vergleich zu langsamen Systemen, etwa mit den Prozessoren 6502 oder 6809, arbeiten moderne Busse asynchron, und periphere Einheiten „kennen" den Prozessorzustand nicht oder nur teilweise. Auch fehlt eine automatische Refresh-Generierung, wie man sie von Z80-Systemen kennt, da diese einer ernsthaften Multiprozessor-Anwendung widerspricht. Daher müssen alle Peripherieeinheiten so konzipiert sein, daß sie ihre Aufgaben unabhängig vom Prozessortyp erfüllen – und das oft am Rande ihrer minimalen Reaktionszeit.

Bei dynamischen RAMs kommt erschwerend der Umstand hinzu, daß jeder Speicherbaustein – z. B. der 256-KBit-Speicher 41256 – im Moment des Zugriffs 100 mA „schluckt"; das sind bei 1 MByte immerhin 3,2 A. Ein ungeschicktes Design bewirkt dadurch eine Art Schockwelle, die sich auf den gesamten Computer auswirken kann. Besondere Anforderungen ergeben sich dabei auch an Speichertestprogramme, die neben einfachen Funktionstests den Einfluß von Resonanzen im Rechner berücksichtigen müssen.

Eine Zugriffskette ohne Wartezyklen

Bei der Entwicklung dynamischer RAM-Karten hat zweifelsfrei die Zugriffszeit auf die Daten vorrangige Bedeutung. Ein MC68000 läßt dem Speicher bei 8 MHz Taktfrequenz genau 375 ns Reaktionszeit; der Rest wird durch Wartezyklen eingeholt. Da gängige 256-KBit-Speicher eine Zugriffszeit von 150 ns garantieren, bleibt der Zugriffslogik nach Abzug von etwa 50 ns Datenlaufzeit durch den Bus nur mehr 175 ns Decodierzeit übrig. Allein die Gatterlaufzeit eines einzigen LS-Flipflops sollte mit 45 ns veranschlagt werden, so daß bei der RAS-MUX-CAS-Kette 135 ns verlorengehen. In der Restzeit von 40 ns muß schließlich die Entscheidung fallen, ob ein Zugriff überhaupt erlaubt ist oder noch ein Refresh im Gange ist und welche der installierten Speicherbänke angesprochen werden sollen. Diese Zeit ist für LS-Logik zu kurz.

Abhilfe schafft der Einsatz von ALS- oder S-Logik. Unbedingt verzichtet werden sollte dagegen auf HC- oder HCT-Typen, da diese aufgrund der hohen Zugriffshäufigkeiten zu viel Strom ziehen und ein ungünstiges Flankenverhalten im gemischten Betrieb mit LS-Logik aufweisen. *Bild 1* zeigt den Aufbau der RAS-MUX-CAS-Kette aus vier Flipflops, die in dieser Beschaltung einen vollständig asynchronen Betrieb ohne Wartezyklen zulassen. Voraussetzung ist allerdings, daß der Refresh nicht während des Zugriffs stattfindet (siehe weiter unten) und synchron zum anliegenden 16-MHz-Takt erfolgt. Die Taktrate des steuernden Prozessortyps ist dagegen frei wählbar. Wichtig ist der Inverter IC 3b, der das Problem von Bauteiletoleranzen ausschließt. Solange die Eingangsdaten von FF 2 auf „H" liegen, ist FF 2 gesperrt. Andernfalls kann es vorkommen, daß FF 1 seine Daten bereits liefert, wenn FF 2 seinen Taktimpuls beantwortet. Langwierige Versuche haben die Notwendigkeit solcher Kniffe nachgewiesen.

Keine Soft-Errors durch entprellte Freigabeleitungen

Unter einem Soft-Error versteht man bei dynamischen RAMs das Löschen einzelner Speicherzellen durch einen vorzeitigen Zugriffsabbruch [2]. Dies rührt daher, daß bei jedem Zugriff die angewählte Reihe an Speicherzellen aufgefrischt wird, d. h. die Speicherkondensato-

ren auf dem Chip werden verstärkt auf- oder abgeladen, je nachdem, ob ihr Ladungsinhalt über oder unter einer gewissen Referenzspannung liegt. Wird nun ein Speicherzugriff zu früh abgebrochen, fällt dieser Vergleich falsch aus, und die entsprechende Reihe wird gelöscht.

In den meisten Speicherschaltungen entstehen Soft-Errors durch den Strom, der im Moment des Zugriffs durch die RAM-Bank fließt. Er bewirkt nämlich nicht allein den Zusammenbruch der Versorgungsspannung; viel unangenehmer ist ein lokales Anheben des Massepotentials im Rechnersystem. Hierdurch wird eine anliegende Leitung mit 0-V-Potential nicht mehr als solche akzeptiert, und nachfolgende Logik-Gatter triggern daraufhin auf gefährliche 2,4 V. Für die angeschlossene RAM-Bank bedeutet das einen vorzeitigen Zugriffsabbruch, der den gefürchteten Datenverlust zur Folge hat.

Die Schaltung in Bild 1 verhindert diesen Abbruch mit Hilfe von FF 1, das die Select-Leitung entprellt. Ist diese einmal aktiv, läßt sich FF 1 erst nach Beendigen des Zugriffs löschen. Hierfür sorgt das ODER-Gatter IC 2a. Auch hier haben zahlreiche Versuche bei einer 2-MByte-RAM-Karte ergeben, daß selbst hochwertige keramische Kondensatoren auf der RAM-Karte zum Abfangen des Zugriffstroms keine Alternative zum Entprellen darstellen. Schaltungen dieser Art sollten auch in anderen schnellen Bus-Systemen Eingang finden.

Arithmetik statt Refresh-Zyklen

Ein weiterer Faktor zur Zeitersparnis stellt ein geschicktes Design zur Refresh-Generierung dar. Alle 60 µs verlangen dynamische RAMs eine Reihenadressierung, gefolgt von einem verkürzten RAS-Impuls, der ihnen die Möglichkeit des Auffrischens ihrer Speicherzellen gewährt. Dabei kommt für moderne 16-Bit-Prozessoren ein versteckter Refresh, bei dem die Refresh-Adressierung nach jedem Prozessorzugriff erfolgt, nicht in Frage: Zum einen sind die Prozessoren so schnell, daß für einen Refresh-Zyklus zwischen den Zugriffen keine Zeit bleibt, zum anderen erfolgen häufig zwei oder mehrere Zugriffe unmittelbar nacheinander, so daß der versteckte Refresh (Hidden Refresh) in diesen Anwendungsfällen die schlechteste Wahl darstellt. Es versteht sich von selbst, daß man dem Prozessor keine absolute Priorität geben kann, da ein forciert abgebrochener Refresh-Zyklus die selben katastrophalen Folgen nach sich zöge, wie sie bei Soft-Errors diskutiert wurden.

Möglichkeiten der Zeitersparnis ergeben sich daher ohne den Einsatz aufwendiger Pre-Prozessoren nur, wenn man die Tatsache nutzt, daß ein Refresh auch bei einem normalen Schreib- oder Lesezugriff erfolgt – und dies für eine ganze Zellenreihe, nicht nur für die aktuell adressierte Speicherzelle. Wenn also jeder Refresh ausgelassen wird, dessen Refresh-Adresse während der erwähnten 60 µs Refresh-Zeit von der CPU angesprochen wurde, kann man in speziellen Anwenderprogrammen sogar ganz ohne Zugriffsverlust arbeiten. Voraussetzung ist nur, daß die CPU schnell genug die nötigen 256 Refresh-Adressen durchläuft.

Bild 2 zeigt die technische Realisierung des Hardware-Komparators, der gleichzeitig die Generierung der Refresh-Adresse und des 60-µs-Zählers enthält. Als Komparatoren sind K 1 und K 2 eingesetzt, die den Stand des Adreßzählers Z 2 mit den gerade anliegenden Reihenadressen am RAM vergleichen. Fällt dieser Vergleich positiv aus (z. B. auch dann, wenn der Adreßzähler selbst die Reihenadresse liefert), werden der Adreß-

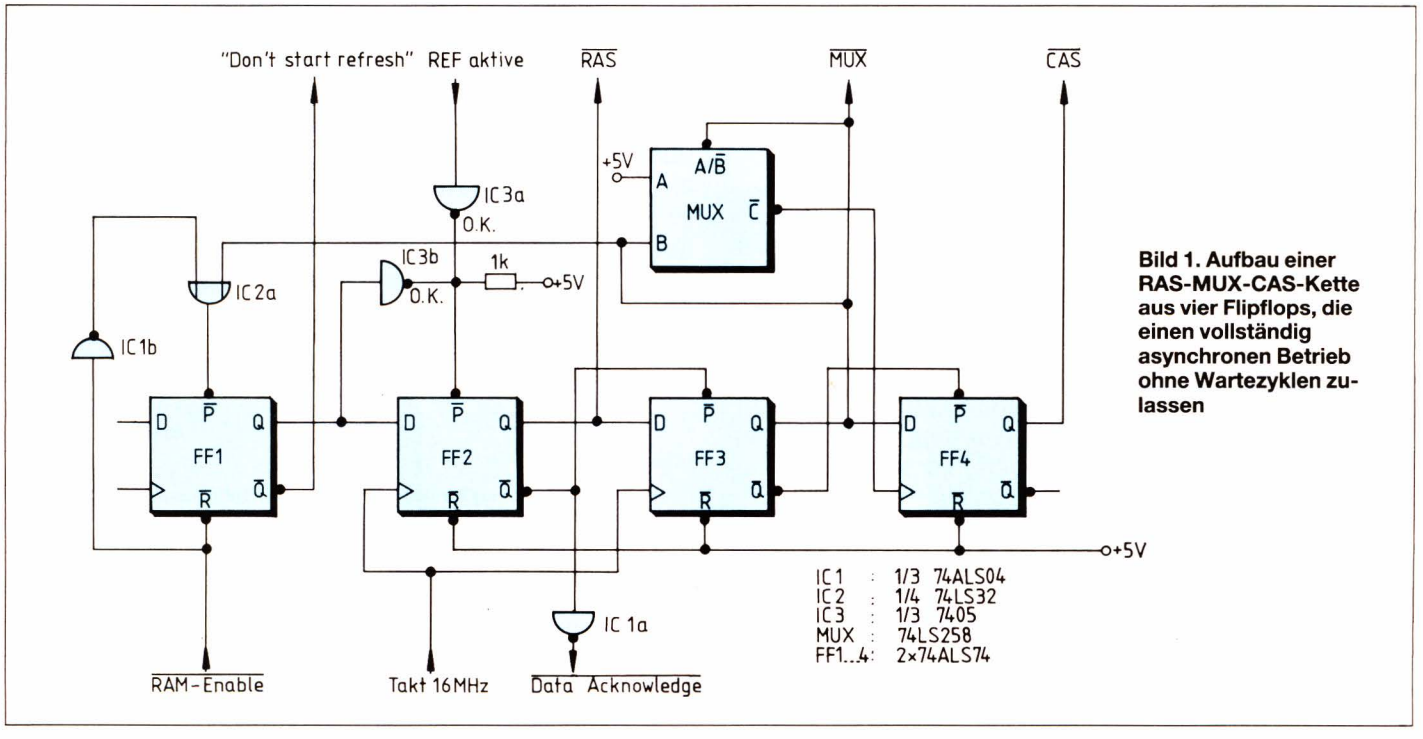


Bild 1. Aufbau einer RAS-MUX-CAS-Kette aus vier Flipflops, die einen vollständig asynchronen Betrieb ohne Wartezyklen zulassen

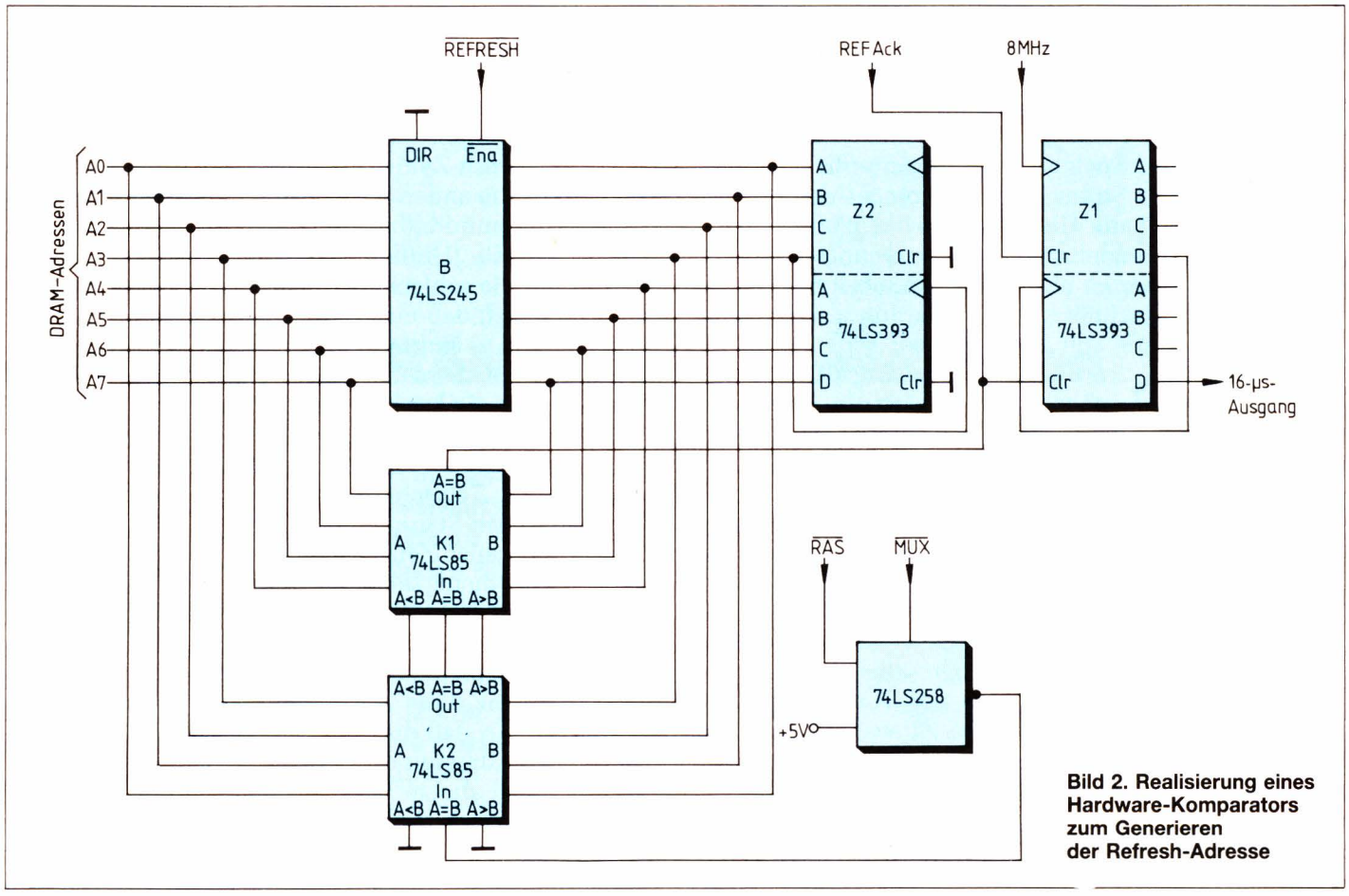


Bild 2. Realisierung eines Hardware-Komparators zum Generieren der Refresh-Adresse

zähler um eins weitergezählt und der 60-µs-Zähler (Z 1) zurückgesetzt. Die Ausgänge von Z 1 können übrigens auch vorteilhaft zum Erzeugen des RAS-Zyklus während des Auffrischens dienen. Dabei sei besonders auf das Einhalten der Precharge-Zeit – einer unbedingt notwendigen Pause zwischen zwei RAS-Zyklen – hingewiesen.

Arithmetik auch beim Selektieren

Ein Problem ganz anderer Art stellt das Selektieren von Massenspeichern dar. Da es günstig ist, den Speicher an jede physikalische Adresse im gesamten Adreßraum legen zu können, empfiehlt sich – einer Idee von Hagen Völzke folgend [3] – der Einsatz von Hardware-Addierern des Typs 74LS83, die zur physikalischen Adresse des Mikrocomputers einen mit Jumpern einstellbaren Wert addieren. Deren Ausgang läßt sich dann in geschickter Weise zum Selektieren und Blockumschalten mehrerer RAM-Bänke nutzen.

Zur Realisation solcher Speicher ist abschließend zu bemerken, daß jedes Speicher-IC mit einem keramischen Kondensator von mindestens 0,47 µF versehen sein muß. Dieser Wert ergibt sich aus dem Strom, den die ICs während eines Zugriffs benötigen. Um ein Schwingen mit niedriger Frequenz zu vermeiden, wie es bei blockweisem Wechsel eines Anwenderprogrammes zwischen mehreren RAM-Bänken entsteht, sollten Tantal-Kondensatoren eingestreut sein.

Schließlich versteht sich von selbst, daß alle Adreß- und Datenleitungen an den Speichern auf 2,94 V terminiert sein müssen. Andernfalls ergeben sich Schwingungen auf den Leitungen, die Lese- und Schreibfehler hervorrufen. Auch dies ist ein Problem, das bei den mit „Gleichstrom" von einem bis zu 4 MHz arbeitenden 8-Bit-Prozessoren unbekannt war.

Andreas Grimm

Literatur

[1] *Grimm, A.*: Megabytes für den mc68000. Erscheint demnächst in der mc.
[2] *Wiegmann, A.*: Dynamische 64-K-RAMs – Technik und Produkte. ELEKTRONIK 1982, H. 25, S. 43...56.
[3] Mündliche Mitteilung von Hagen Völzke (Autor in mc).

Einchip-Computer steuert Frequenzzähler

Mit Hilfe von Einchip-Mikrocomputern ist es möglich, einen großen Teil der Funktionen eines Frequenzzählers zu steuern. Vorteilhaft wirkt sich dabei die Vereinfachung der Hardware aus, darüber hinaus lassen sich zusätzliche Funktionen leicht realisieren. Es ist allerdings nicht besonders sinnvoll, einen Einchip-Computer direkt zum Zählen elektrischer Impulse zu verwenden, wenn sie eine Frequenz von mehr als 100 kHz aufweisen. Daher wurde hier der Einchip-Computer als Steuerelement an schnelle bipolare Zählerbausteine angeschlossen, so daß sich insgesamt eine sehr preisgünstige Hardwarelösung ergibt. Als Einchip-Computer lassen sich die CMOS- oder HMOS-Typen aus der 6805-Familie verwenden.

Frequenzzähler bestehen aus digitalen Teilerstufen, die das serielle Eingangssignal in BCD-Zahlen umsetzen. Diese werden mit Hilfe von Zwischenspeichern, Decodern und Treibern an 7-Segment-Displays angelegt. Die Funktion der Frequenzbestimmung übernimmt ein Gatter am Eingang, das von einer Zeitbasis angesteuert wird. Je stabiler und genauer dieses Zeitsignal ist, um so genauer ist auch das Meßergebnis des gesamten Frequenzzählers.

In einem Frequenzzähler, der zur Steuerung einen Einchip-Computers vom Typ M6805 (HMOS) oder M146805 (CMOS) verwendet, läßt sich die Zeit entwe-

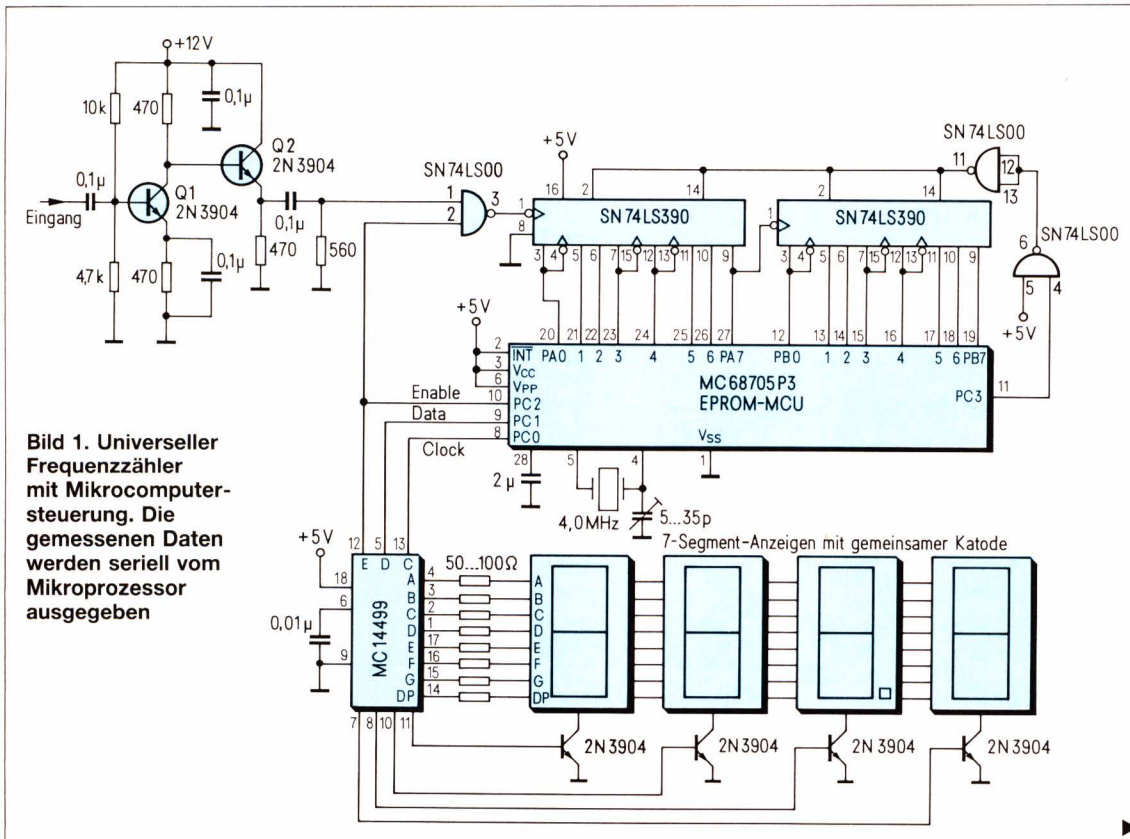


Bild 1. Universeller Frequenzzähler mit Mikrocomputersteuerung. Die gemessenen Daten werden seriell vom Mikroprozessor ausgegeben

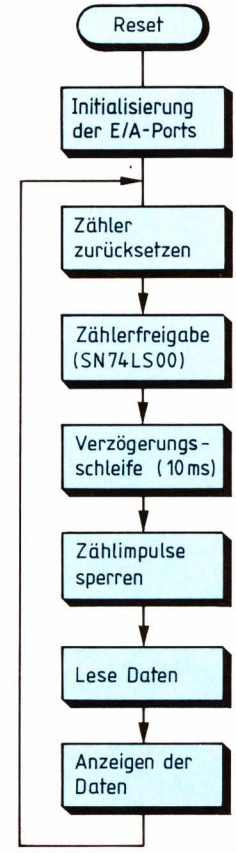


Bild 2. Flußdiagramm des Programms für den Frequenzzähler

der mit Hilfe eines internen Timers oder mit einer Software-Zeitgeber-Schleife berechnen. Auch in diesem Fall wird die Genauigkeit des Zählers durch den Grad der Genauigkeit der Zeitsignalerzeugung bestimmt. Am Ende der Zählzeit liest der Einchip-Computer die BCD-Daten der Dezimalzähler (*Bild 1*) über seine Portanschlüsse (PA0...PA7 und PB0...PB7). Anschließend werden diese Daten seriell an den LED-Treiber MC14499 weitergeleitet. Dieser Baustein kann bis zu vier 7-Segment-Displays mit gemeinsamer Katode ansteuern. Der Hardware-Aufwand ist sehr gering. Für die Anzeigeeinheit sind lediglich acht Widerstände zur Strombegrenzung und vier Transistoren für das Schalten des Katodenstroms erforderlich.

Geringer Hardwareaufwand

Am Eingang des Zählers befindet sich eine Verstärker- und Signalformer-Stufe (Transistoren Q1 und Q2). Das Zeitsignal öffnet und schließt das NAND-Gatter SN74LS00, an dessen Ausgang die beiden in Reihe liegenden Dekadenzähler SN74LS390 angeschlossen sind. In dem hier gezeigten Schaltungsbeispiel kommt als Steuereinheit die EPROM-MCU MC68705P3 zur Anwendung, die den Einchip-Computer MC6805P2 emuliert. Die vier BCD-Ausgänge der Dekaden liegen an den A- bzw. B-Ports des Einchip-Computers. Dargestellt werden auf dem 7-Segment-Display die Dekaden 100 kHz, 10 kHz, 1 kHz und 100 Hz. Wenn der Anzeigeumfang größer sein soll (z. B. die Werte für 10 Hz und 1 Hz auch anzuzeigen sind), müssen zusätzliche Dekadenzähler und E/A-Anschlüsse benutzt werden. Es besteht auch die Möglichkeit, mit Hilfe von Tri-State-Multiplexern die Zahl der erforderlichen E/A-Anschlüsse zu reduzieren.

Durch Veränderung der Torzeit läßt sich auch das Anzeigefenster verschieben. So zeigt z. B. bei einer Torzeit von 1 ms die linke Anzeigeeinheit den Wert für MHz an. In der Regel ist eine vierstellige Anzeige ausreichend, z. B. wenn die Empfangsfrequenz eines Radios wiedergegeben werden soll.

Genauigkeit von der Zeitbasis abhängig

Wie bei allen Zählerschaltungen dieser Art, bestimmt die Qualität der Zeitbasis die Genauigkeit des Gerätes. Im hier vorgestellten Schaltungsbeispiel bestimmt ein Quarz und ein Trimmkondensator die Taktfrequenz der

```
00001                                NAM     CNTR
00002                        *
00003                        *
00004          0000        A PORTA   EQU     $000
00005          0001        A PORTB   EQU     $001
00006          0002        A PORTC   EQU     $002
00007                        *
00008          0004        A PADDR   EQU     $004
00009          0005        A PBDDR   EQU     $005
00010          0006        A PCDDR   EQU     $006
00011                        *
00012          0040        A SAVEA   EQU     $040
00013          0041        A SAVEB   EQU     $041
00014          0042        A TEST    EQU     $042
00015                        *
00016A 0784                          ORG     $784
00017A 0784    07          A MOR     FCB     $07        XTAL MODE
00018                        *
00019                        *  BEGINNING OF CODE
00020                        *
00021A 0080                          ORG     $80
00022                        *
00023A 0080 3F 04          A START   CLR     PADDR      ALL INPUTS
00024A 0082 3F 05          A         CLR     PBDDR      HERE TOO
00025A 0084 A6 0F          A         LDA     #$0F       ALL OUTS
00026A 0086 B7 06          A         STA     PCDDR
00027                        *
00028A 0088 16 02          A CONTIN  BSET    3,PORTC    CLEAR OUT COUNTERS
00029A 008A 17 02          A         BCLR    3,PORTC
00030                        *
00031A 008C 14 02          A         BSET    2,PORTC    ENABLE COUNTER
00032                        *
00033                        *  BEGINNING OF TIMING LOOP
00034                        *
00035A 008E A6 ED          A         LDA     #$ED
00036A 0090 9D               LOOP    NOP
00037A 0091 9D                       NOP
00038A 0092 3A 42          A         DEC     TEST
00039A 0094 3A 42          A         DEC     TEST
00040A 0096 3A 42          A         DEC     TEST
00041A 0098 3C 42          A         INC     TEST
00042A 009A 3C 42          A         INC     TEST
00043A 009C 4A                       DECA
00044A 009D 26 F1     0090           BNE     LOOP
00045                        *
00046A 009F 3C 42          A         INC     TEST
00047A 00A1 3C 42          A         INC     TEST
00048A 00A3 3C 42          A         INC     TEST
00049A 00A5 3C 42          A         INC     TEST
00050A 00A7 3C 42          A         INC     TEST
00051A 00A9 9D                       NOP
00052                        *
00053                        *  FINALLY ALL COUNTED
00054                        *
00055A 00AA 15 02          A         BCLR    2,PORTC    HOLD OFF ANY COUNTS
00056                        *
00057                        *  NOW WAITED 10 MS
00058                        *
00059A 00AC B6 00          A         LDA     PORTA      GET LS DIGITS
00060A 00AE B7 40          A         STA     SAVEA
00061                        *
00062A 00B0 B6 01          A         LDA     PORTB      GET MS DIGITS
00063A 00B2 B7 41          A         STA     SAVEB
00064                        *
00065                        *  SEND OUT DATA NOW
00066                        *
00067A 00B4 15 02          A SNDDTA  BCLR    2,PORTC    ENABLE = 0
00068A 00B6 10 02          A         BSET    0,PORTC    CLOCK = 1
00069                        *
00070A 00B8 AE 05          A         LDX     #$05       NUMBER TO ROTATE
00071A 00BA A6 20          A         LDA     #$20       DECIMAL POINT HERE
00072A 00BC AD 0E     00CC           BSR     CLOCK1
00073                        *
00074A 00BE B6 41          A         LDA     SAVEB      GET LSBS FIRST
00075A 00C0 AD 08     00CA           BSR     CLOCK
00076                        *
00077A 00C2 B6 40          A         LDA     SAVEA      NOW MSB
00078A 00C4 AD 04     00CA           BSR     CLOCK
00079                        *
00080A 00C6 14 02          A         BSET    2,PORTC    NOW ENABLE IT
00081                        *
00082A 00C8 20 BE     0088           BRA     CONTIN     CONTINUE COUNTING
00083                        *
00084                        *
00085A 00CA AE 08          A CLOCK   LDX     #$08
00086A 00CC 49               CLOCK1  ROLA
00087A 00CD 24 0A     00D9           BCC     ZERO
00088A 00CF 12 02          A         BSET    1,PORTC
00089A 00D1 11 02          A CLOCK2  BCLR    0,PORTC    SHIFT IN ONE BIT
00090A 00D3 10 02          A         BSET    0,PORTC
00091A 00D5 5A                       DECX
00092A 00D6 26 F4     00CC           BNE     CLOCK1
00093A 00D8 81                       RTS
00094                        *
00095A 00D9 13 02          A ZERO    BCLR    1,PORTC    MAKE A ZERO
00096A 00DB 20 F4     00D1           BRA     CLOCK2     GO CLOCK IT
00097                        *
00098                        *
00099                        *  RESTART VECTORS
00100                        *
00101                        *
00102A 07F8                          ORG     $7F8
00103                        *
00104A 07F8          0080    A TIMER   FDB     START
00105A 07FA          0080    A EXTINT  FDB     START
00106A 07FC          0080    A SWIINT  FDB     START
00107A 07FE          0080    A RESET   FDB     START
00108                        *
00109                                  END
```

Bild 3. Programmlisting. Für die Frequenzzählerfunktion sind lediglich etwa 256 Byte ROM erforderlich

MCU und die Torzeit. Der interne Timer oder die Software-Timer-Schleife teilt diese Frequenz in einem festen Verhältnis herunter, der Trimmer ermöglicht eine Feineinstellung der Zeitbasis. Zum Abgleich ist lediglich ein Signal mit bekannter Frequenz notwendig, das am Eingang des Zählers anliegt. Anschließend wird der Trimmer entsprechend vorgestellt, bis der Anzeigenwert mit der Kalibrierfrequenz übereinstimmt.

Nur 256 Byte Software

Für die beschriebene Schaltung ist relativ wenig Software erforderlich. Es bereitet keine Probleme, das Programm in einem der erwähnten Einchip-Computerbausteine unterzubringen. Der Typ MC68705P3 bietet z. B. 1,8 KByte EPROM und der Typ MC6805P2 1 KByte ROM. Benötigt werden allerdings nur etwa 256 Byte ROM bzw. EPROM. Das Flußdiagramm für die Zählerroutine zeigt *Bild 2* und das Programmlisting ist in *Bild 3* dargestellt. Nicht enthalten im Programm ist eine Maßnahme zur Unterdrückung des Flackerns der letzten Stelle. Diese läßt sich leicht realisieren, indem man den Mittelwert von acht Messungen bildet. Dazu werden acht Messungen vorgenommen, addiert und anschließend durch acht dividiert (geteilt durch zwei, geteilt durch zwei, geteilt durch zwei).

Der Programmiervorgang des Einchip-Mikrocomputers MC68705P3 ist in [1] beschrieben. Andere Typen der 6805-Familie, z. B. die HMOS- und CMOS-Bausteine, besitzen mehr E/A-Leitungen, als es für den Frequenzzähler erforderlich ist. Damit ist es möglich, weitere Funktionen in dem Meßgerät unterzubringen. Beispiel dafür ist eine Spannungsmesserfunktion, wenn der Baustein MC6805R2 Verwendung findet, der über einen A/D-Umsetzer verfügt.

Tim Ahrens

Literatur

[1] Applikationsbericht AN-857 der Firma Motorola.

Mit einfachem Zeitgeber-Baustein:

Resetlogik, Spannungsüberwachung und Watchdog

Mikroprozessorsysteme benötigen nach dem Einschalten der Versorgungsspannung einen definierten Resetimpuls. Dafür sorgt in vielen Fällen der Zeitgeber-Baustein 555. Mit wenig Aufwand läßt sich die Resetlogik so erweitern, daß gleichzeitig der Spannungspegel und die einwandfreie Funktion der Software überwacht werden.

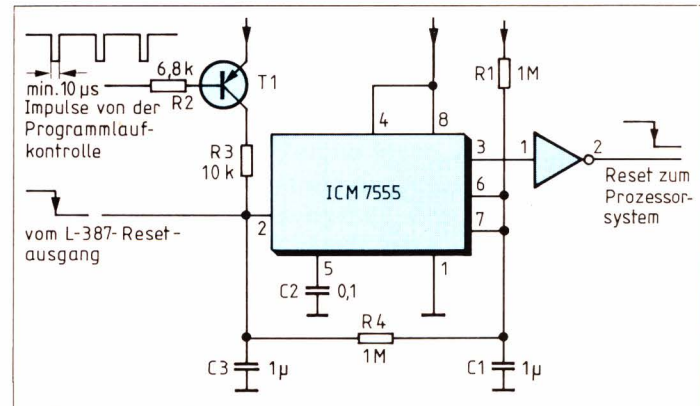
Voraussetzung für die Spannungsüberwachung ist der Reglerbaustein L 387 (SGS), der beim Absinken der Eingangsspannung einen negativen Impuls liefert. Der ordnungsgemäße Programmablauf wird durch eine sogenannte Watchdog-Funktion gewährleistet. Dabei hat das Programm periodisch Impulse auszugeben. Bleiben sie aus (wegen einer Störung), wird ein Reset ausgelöst.

Die Schaltung (*Bild*) ist mit dem CMOS-Zeitgeber ICM 7555 aufgebaut. Beide internen Komparatoren werden benutzt. Die RC-Kombination R 1/C 1 erzeugt mit einem der internen Komparatoren den zeitlich definierten Resetimpuls beim Einschalten der Betriebsspannung.

An die Basis des PNP-Transistors gelangen die negativen Impulse von der Programmlaufkontrolle. Die zulässige Zeit zwischen den einzelnen Impulsen läßt sich mit R 3/C 4 einstellen. Bleiben sie aus (Absturz des Prozessors), entlädt sich die positive Triggerspannung am Zeitgebereingang 2 über den Widerstand R 4, und der zweite interne Komparator kippt. Das Mikroprozessorsystem erhält einen definierten Resetimpuls.

Der Resetausgang (Open Collector) des Spannungsreglers L 387 ist direkt mit dem Triggereingang 2 verbunden. Sinkt die Versorgungsspannung unter 5,5 V, wird der Resetausgang des L 387 aktiviert (aktiv-low), und der Zeitgeber erzeugt ebenfalls einen Reset.

Gerd Radzewitz



Resetlogik, Spannungsüberwachung und Watchdog mit wenig Aufwand realisiert

Für Entwicklung und Service:

6809-Systeme preiswert getestet

Ohne Eingriff in die Hardware kommt das Testgerät aus, das wir Ihnen in diesem Beitrag vorstellen. Es wird über die Fassung der CPU an das zu prüfende System angekoppelt. Ab einer frei wählbaren Triggeradresse hält es den Prozessor an und schaltet auf Einzelschrittbetrieb um.

Das Testgerät stoppt den Prozessor, indem es spätestens im vorletzten Zyklus eines Befehls einen Low-Pegel an den $\overline{\text{HALT}}$-Eingang der CPU sendet. Dieser Low-Pegel wird vom Testgerät erzeugt, wenn die eingestellte Triggeradresse mit der Adresse auf dem Adressenbus des Prozessorsystems übereinstimmt. Die CPU hält dann am Ende des gerade laufenden Befehls an und

Das Testgerät (*Bild 1*) beschränkt sich auf die Überprüfung weniger Grundfunktionen des Prozessors. Es erfüllt folgende Anforderungen:

- Boolesche Triggerung auf Adressensignale ermöglicht einen Prozessorstop.
- Anschließende Einzelschrittabarbeitung des Programms mit Analyse des kompletten *nächsten* Befehls. Angezeigt werden die Adressenbytes, das Datenbyte sowie die wichtigsten Steuersignale.
- Testbarkeit von Prozessorsystemen, in denen dynamische Schreib-/Lese-Speicher eingebaut sind.
- Ankopplung an das 6809-System über einen Zwischenstecker, der in die Fassung der CPU gesteckt wird (*Bild 2*).

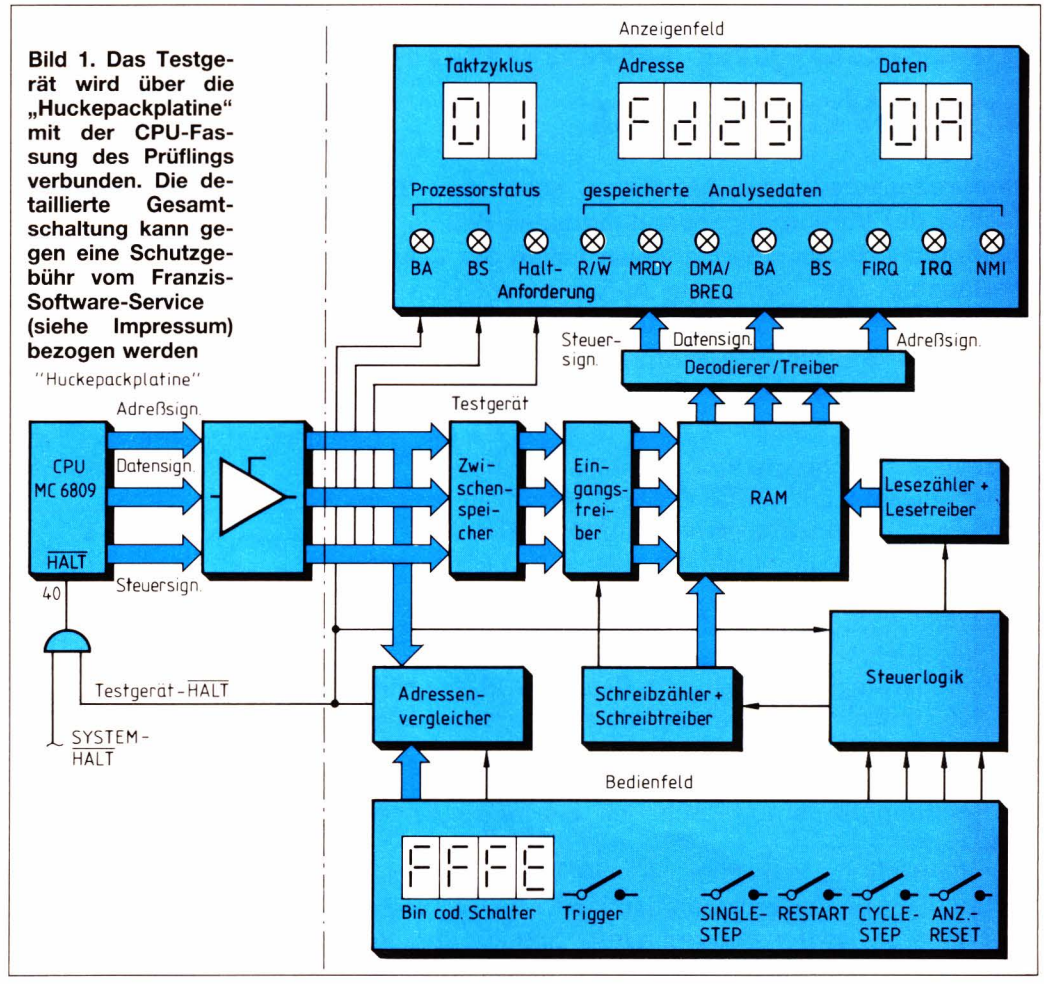


Bild 1. Das Testgerät wird über die „Huckepackplatine" mit der CPU-Fassung des Prüflings verbunden. Die detaillierte Gesamtschaltung kann gegen eine Schutzgebühr vom Franzis-Software-Service (siehe Impressum) bezogen werden

schaltet ihre Busse in den hochohmigen Zustand. Solange ein Low-Pegel am $\overline{HALT}$-Eingang anliegt, verharrt die CPU in diesem Zustand, ohne den Verlust der in den Registern gespeicherten Daten. Den $\overline{HALT}$-Zustand kennzeichnet die CPU durch die Statussignale BA=BS=1.

Danach erfolgt eine Einzelschrittabarbeitung der folgenden Befehle, die dadurch ermöglicht wird, daß der Low-Pegel am $\overline{HALT}$-Eingang für mindestens eine Q-Taktperiode aufgehoben wird. Der Prozessor bearbeitet den nächsten Befehl und hält nach dessen Beendigung wieder an. Dieser Befehl, der bis zu 20 Taktzyklen lang sein kann, wird in den Speicher des Testgerätes eingeschrieben. Die in diesem Befehl aufgetretenen Adressen und Daten können nacheinander auf einem Hexadezimal-Display dargestellt werden; Steuersignale werden von LEDs angezeigt.

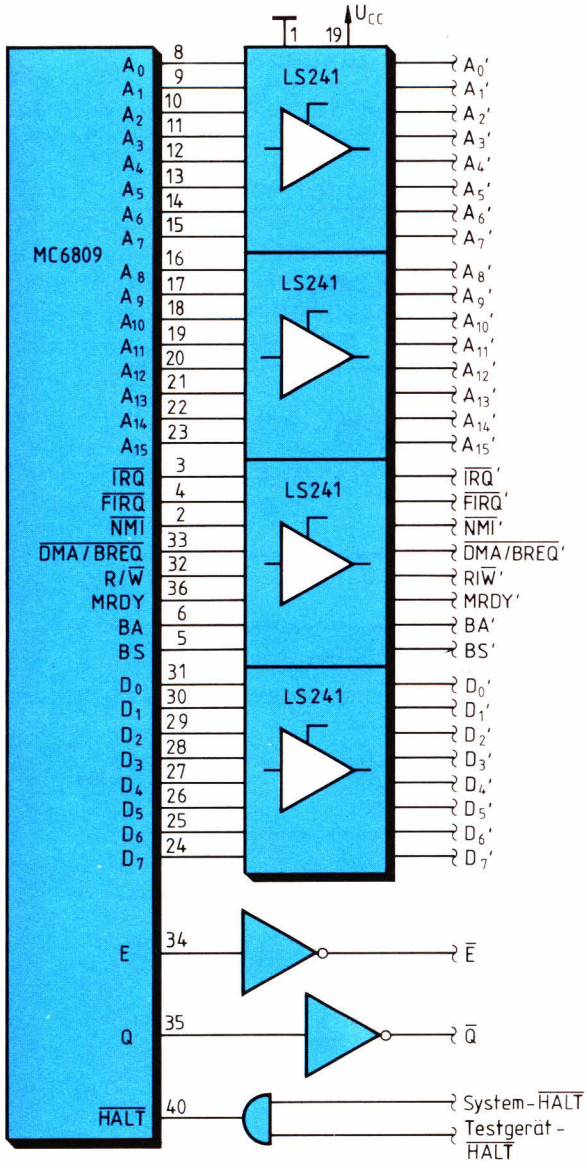


Bild 2. Neben der CPU befinden sich auf der „Huckepackplatine" die Treiber für Daten-, Adreß- und Steuersignale

Während der Phase, in der das Taktsignal E=0 ist, wird der Datenbus von der CPU nicht benutzt. Die meisten Systeme benutzen diese Phase, um dynamische RAMs aufzufrischen. Da während des $\overline{HALT}$-Zustands die Taktsignale E und Q normal weiterlaufen, ist es möglich, Prozessorsysteme zu überprüfen, in denen dynamische RAMs eingebaut sind.

1 Befehlsabarbeitung

Zum Verständnis der Analyse soll im folgenden die Befehlsabarbeitung der CPU 6809 erklärt werden: Ein Befehlszyklus ist unterteilt in mehrere Taktzyklen, die mit der fallenden E-Taktflanke beginnen und mit der folgenden negativen enden. Für einen Befehlszyklus werden minimal zwei Taktzyklen und maximal 20 Taktzyklen benötigt.

Im ersten Zyklus liest die CPU einen Op-Code in das Befehlsregister ein. Im zweiten Zyklus wird das nächste Programmbyte gelesen, denn die meisten Befehle benötigen ein solches. Zur gleichen Zeit decodiert die CPU den Op-Code und erkennt daran, wie viele Zyklen zur Abarbeitung des Befehls nötig sind.

Es gibt nun aber eine ganze Reihe von Befehlen, bei denen nicht direkt mit dem ersten Byte die Länge des Befehls feststeht. Diese Fälle werden ausführlich in [1] beschrieben.

Wollte man jeden Befehl mit seiner *ihm eigenen* Taktzyklenzahl erfassen, wäre eine umfangreiche Befehlsdecodierung notwendig. Da die längsten Befehle, CWAI, SWI/2 und SWI/3, 20 Zyklen beanspruchen, ist es einfacher, bei der Einzelschrittabarbeitung des Programms jedesmal 20 Zyklen, beginnend mit dem Op-Code-Holzyklus, zu speichern und Zyklus für Zyklus zur Anzeige zu bringen. Die „umsonst" gespeicherten Zyklen werden vom Anwender bei der Analyse nicht berücksichtigt.

2 Die Analyse

Die Haltanforderung vom Testgerät wird über eine LED sichtbar gemacht. Am Ende des letzten Zyklus bringt der Prozessor seine Statusleitungen BA und BS in den aktiven Zustand, was von zwei weiteren LEDs angezeigt wird.

Durch Drücken der Taste „Singlestep" wird die Haltanforderung für die Dauer zweier Q-Taktperioden aufgehoben. Der Prozessor bearbeitet den im Programm folgenden Befehl und hält wieder an. Dabei gelangen die in D-Flipflops am Ende eines jeden Zyklus zwischengespeicherten Daten über die durchgeschalteten Eingangstreiber an die Dateneingänge der RAMs, die in den Schreib-Modus gebracht worden sind. Wenn die Daten des ersten Zyklus an den Dateneingängen der RAMs anstehen, legt der Schreibzähler mit der fallenden E-Flanke die Adresse 1_{dez} über einen ebenfalls durchgeschalteten Tri-State-Treiber an die Adresseneingänge A0...A4 der RAMs. Das invertierte Q-Taktsignal wird dem $\overline{CS1}$-Eingang zugeführt und dient dem Einschreiben der Daten in die RAMs. Der Schreibzähler wird mit

jeder fallenden E-Flanke um eins hochgezählt, bis er beim Erreichen von 21_{dez} über seinen $\overline{CE}$-Eingang gestoppt wird. Damit sind 20 Zyklen, beginnend mit der Op-Code-Holphase, in die RAMs eingeschrieben worden.

Die fallende E-Flanke, die den Schreibzähler stoppt und den Schreib- und die Eingangstreiber in den hochohmigen Zustand bringt, schaltet den Lesetreiber durch, über den der Lesezähler die Adresse 01_{hex} an die Adresseneingänge der RAMs legt. Gleichzeitig gelangen die RAMs in den Lese-Modus. Über den Lesetreiber gelangt ein $\overline{CS1}$-Signal an die RAMs, das aus den negierten konjugierten Taktsignalen gebildet wird. Über die als Decodierer/Treiber geschalteten CMOS-Bausteine MC14495 gelangen Adressen- und Datensignale des ersten Zyklus des gerade beendeten Befehls an die Anzeigeeinheiten. Die Steuersignale werden ständig von der negativen Q-Taktflanke in einem D-Flipflop gespeichert (*Bild 3*).

Beim Betätigen der Taste „Cyclestep" gelangt mit der fallenden E-Flanke ein positiver Nadelimpuls an den Takteingang des Lesezählers. Dadurch wird dieser um eins hochgezählt und adressiert den nächsten Zyklus. In der Taktzyklenanzeige erscheint 02_{hex}, und auf den anderen Anzeigen erscheinen die Daten des zweiten Zyklus. So kann durch mehrmaliges Betätigen der Taste „Cyclestep" der Befehl Zyklus für Zyklus ausgelesen werden. Da die meisten Befehle weniger als 20 Taktzyklen lang sind, werden die „zuviel" gespeicherten Zyklen bei der Analyse nicht berücksichtigt.

Ein Aufleuchten der Leuchtdioden BA und BS, die am Steuersignalspeicher angeschlossen sind, zeigt den ersten „umsonst" gespeicherten Zyklus an. Wenn der Lesezähler bis 20_{dez} hochgezählt worden ist, wird er gestoppt. Ein weiteres Betätigen der Taste „Cyclestep" hat somit keinen Einfluß auf den Lesezähler.

Mit der Taste „Anz. Reset" wird der Lesezähler auf 01_{hex} zurückgesetzt, und der letzte Befehl kann noch einmal Zyklus für Zyklus durchgetastet werden.

Durch Drücken der Taste „Restart" wird der Prozessor in den normalen Arbeitsablauf zurückversetzt. In der Anzeige erscheinen Nullen, und die Leuchtdioden erlöschen.

3 Ankopplung des Testgerätes

Zur Ankopplung an das 6809-System wird in den Sockel für die CPU ein Zwischenstecker eingesetzt, der

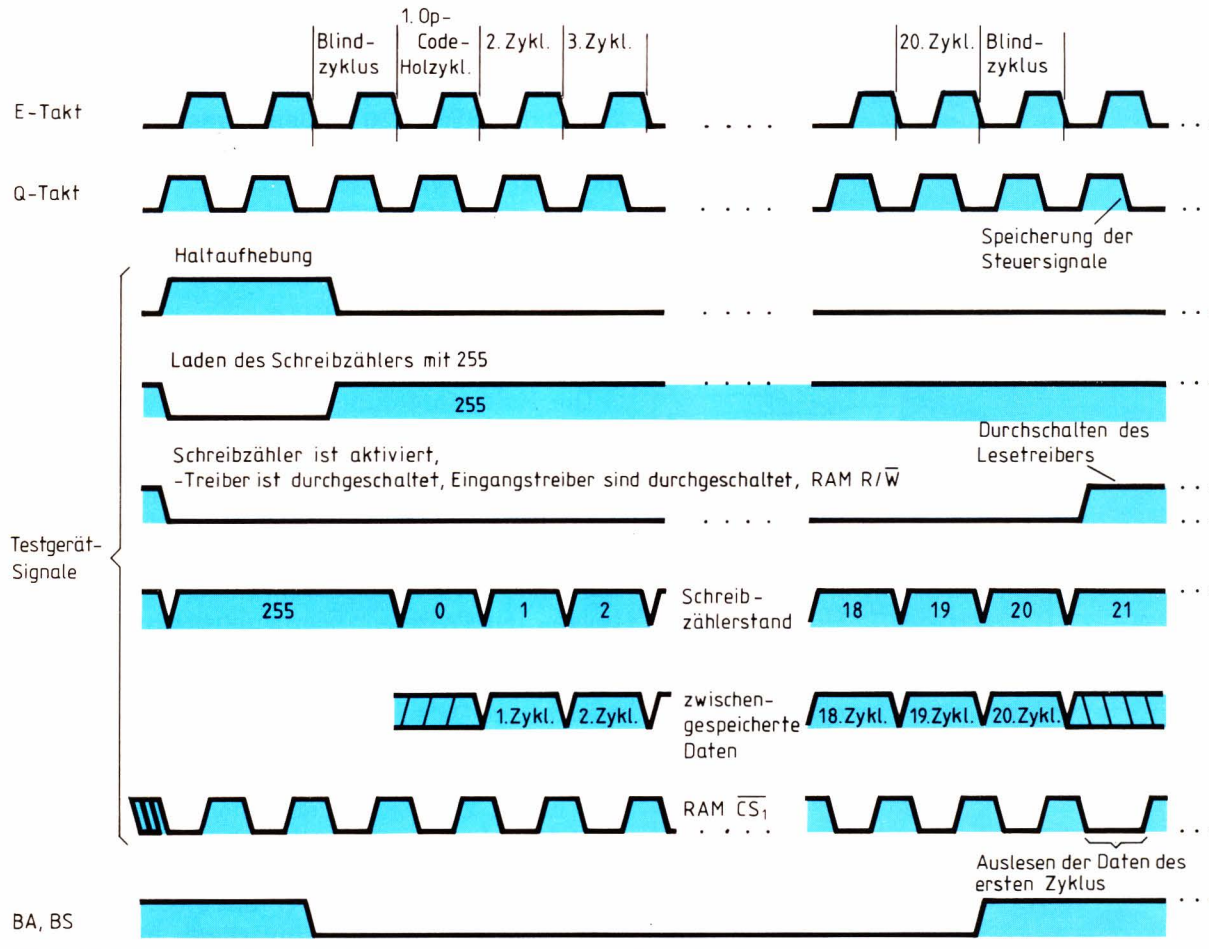


Bild 3. Zeitdiagramm für den Erfassungsvorgang und anschließendes Lesen des Speichers

auf einer kleinen Platine eingelötet ist (Bild 2). Diese „Huckepackplatine" nimmt die CPU und die LS-TTL-Treiber auf. Damit der Prozessor sowohl vom Anwendersystem als auch vom Testgerät angehalten werden kann, ist eine UND-Verknüpfung der Signale „System-$\overline{\text{HALT}}$" und „Testgerät-$\overline{\text{HALT}}$" durchgeführt worden.

Alle Gatter auf der „Huckepackplatine" werden vom Anwendersystem mit Spannung versorgt. Die gepufferten Signale führen über ein 40poliges Flachbandkabel in das Testgerät.

Torsten Grüning, Dipl.-Ing. Walter Kamm

Literatur

[1] *Safferthal, R.*: Logikanalyse an 6809-Systemen, ELEKTRONIK 1981, H. 25/26, S. 71...76.

[2] MC6809-Datenblatt (1979), ADI-804, Motorola.

Komplexe Multiplikation in kurzer Zeit

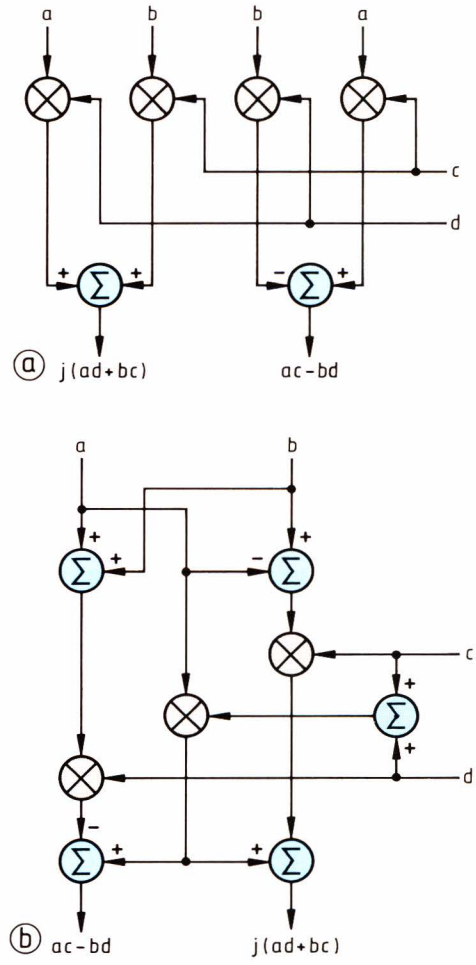
Bei vielen Grafik- und Signalverarbeitungs-Algorithmen sind komplexe Multiplikations-Operationen erforderlich. Wenn diese Aufgabe mit einer flexiblen MPU wie dem Typ 68000 gelöst wird, kann die erforderliche Zeit relativ lang werden, denn für eine komplexe Multiplikation muß man vier Einzelmultiplikationen ausführen. Jede Multiplikation benötigt bis zu 70 Taktzyklen. Merklich verkürzen läßt sich die Multiplikationszeit mit Hilfe eines einfachen Algorithmus. Die komplexe Multiplikation kann so neu strukturiert werden, daß lediglich drei Einzelmultiplikationen durchzuführen sind (*Bild*). Als Ergebnis zeigt sich eine Zeitersparnis von 18 % im Vergleich zur konventionellen Methode. Der Algorithmus lautet:

$$(a + jb) \cdot (c + jd) = [a(c + d) - (a + b)d] + j[c(b - a) + (c + d)\,a] = (ac - bd) + j\,(ad + bc)$$

Randy Wilhelm

Tabelle der abzuarbeitenden Befehle

konventionell			modifizierte Multiplikation		
MOVEM.W	(A1)+, D1...D4	28	MOVEM.W	(A1)+, D1...D4	(28)
A1 zeigt auf den Speicherplatz von a. b, c und d befinden sich in den darauffolgenden Plätzen					
MOVE .W	D1, D5	(4)	MOVE .W	D1, D5	(4)
MOVE .W	D2, D6	(4)	MOVE .W	D3, D6	(4)
MULS	D4, D1	(70)	ADD .W	D2, D1	(4)
MULS	D4, D2	(70)	SUB .W	D5, D2	(4)
MULS	D3, D5	(70)	ADD .W	D4, D3	(4)
MULS	D3, D6	(70)	MULS	D4, D1	(70)
ADD .L	D1, D6	(4)	MULS	D5, D3	(70)
SUB .L	D2, D5	(4)	MULS	D6, D2	(70)
			ADD .L	D3, D2	(4)
			SUB.L	D1, D3	(4)
Anzahl der Taktzyklen: 324			Anzahl der Taktzyklen: 266		



Mit einer mathematischen Umstellung ergibt sich ein Multiplikationsalgorithmus für komplexe Zahlen, der mit nur drei Einzelmultiplikationen auskommt; a) zeigt die konventionelle Methode, b) zeigt den minimierten Aufbau

Verkürzter WAIT-Zyklus beschleunigt 68000-Systeme

Daten-, Adreß- und Steuersignale des Mikroprozessors 68000 sind mit dem Prozessortakt starr gekoppelt. Sind die Peripheriebausteine nicht schnell genug, dann fügt die CPU einige WAIT-Zyklen in den normalen Ablauf ein. Dadurch sinkt die Leistung des Systems. Dieser Beitrag zeigt, daß man die Leistungseinbuße mit einer kleinen Zusatzschaltung in Grenzen halten kann.

Nachdem die Peripherie Adresse und Datum erkannt hat, bestätigt sie das durch die abfallende Flanke des Signals $\overline{\text{DTACK}}$ (*Bild 1*). Geschieht das nicht mindestens um die sogenannte Set-up-Zeit (t_{ASI} = 20 ns bei 8 MHz) vor der abfallenden Flanke von S4, dann fügt der Prozessor WAIT-Zyklen ein. Welche Auswirkungen das auf die maximale Betriebsfrequenz hat, zeigt *Tabelle 1*.

Ein normaler Lesezyklus ist beim 68000 vier Taktperioden lang. Der Schreibzyklus dauert fünf Taktperi-

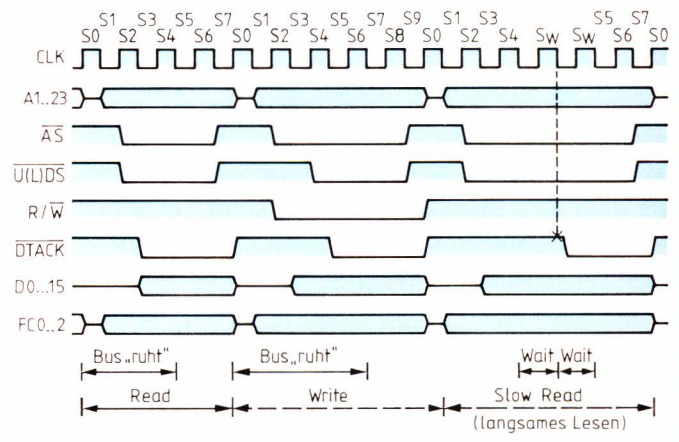


Bild 1. Zeitdiagramm für das Lesen und Schreiben: Bei langsamen Peripheriebausteinen werden WAIT-Zyklen eingefügt

Tabelle 1. Maximale Betriebsfrequenz bei unterschiedlichen RAM- und Treiberbausteinen

RAM-Zugriffszeit t/ns	50	100	150	200	200	250	250	300	350	400	450
Treiber Schottky	28	28	28	28		28					
Treiber LS					56		56	56	56	56	56
$t_{AS/RAS}$	7	7	7	7	22	7	22	22	22	22	22
Verzögerung MC68000 (t_{ASI} & Verz. AS) 10 MHz: 10 + 50 ns	60	60	60								
8 MHz: 15 + 55 ns				70	70	70					
6 MHz: 25 + 65 ns							90	90			
4 MHz: 30 + 75 ns									105	105	105
BUS „ruht" (ns)	145	195	245	305	348	355	418	468	533	583	633
max. Frequenz/MHz (ohne WAIT)	17	12,8	10,2	8,19	7,18	7,04	5,98	5,34	4,69	4,2	3,94
Betriebsfrequenz/MHz (ohne WAIT)	16	12	10	8	7	7	6	5	4	4	3,58

Tabelle 2. Leistungsparameter bei unterschiedlichen RAM- und Treiberbausteinen, bezogen auf die Befehlssequenz „Schreiben-Lesen"

Lösungsweg	Takt-Zyklen	Zeitbedarf	Taktfrequenz	Leistung	Konfiguration
Befehlssequenz (ideal)	17	2125 ns	8 MHz	100 %	200-ns-RAM + Schottky
WAIT bei READ (real)	20	2500 ns	8 MHz	85 %	200-ns-RAM + LS
Reduzierte Taktfrequenz	17	2429 ns	7 MHz	87,5 %	250-ns-RAM + Schottky
Mit Taktdehnung gemäß Bild 3	18,5	2313 ns	8 MHz	92 %	200-ns-RAM + LS

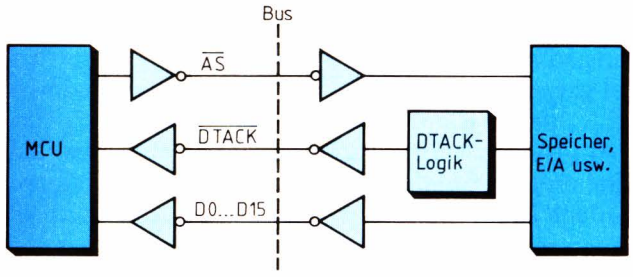


Bild 2. Kopplung des Prozessors mit der Peripherie, vereinfacht dargestellt

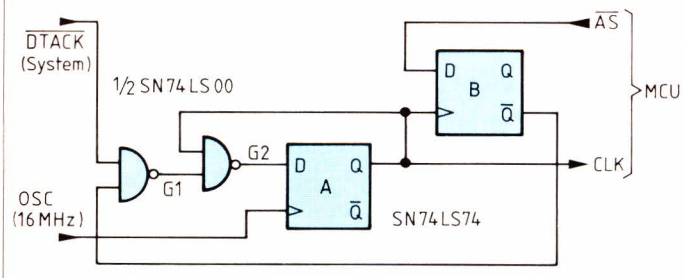


Bild 3. Mit dieser Zusatzschaltung wird der Takt für den Prozessor bei Bedarf gedehnt (S4)

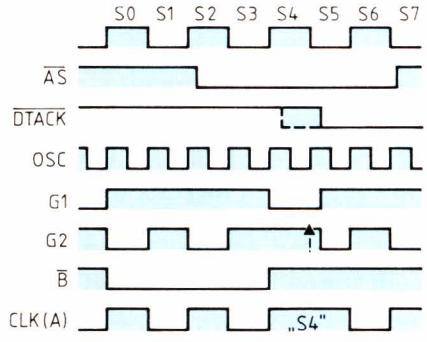


Bild 4. Zeitdiagramm zu Bild 3: Der gedehnte Takt verhindert WAIT-Zyklen und hält so die Leistungseinbuße bei langsamen Peripheriebausteinen in Grenzen

oden. So benötigt die einfachste Befehlssequenz „Lesen-Schreiben" (entspricht einem Datentransport) einschließlich dem Holen der Befehle insgesamt vier Buszugriffe. Darin ist nur ein einziger Schreibvorgang enthalten. Diese einfache Sequenz dauert somit sage und schreibe 17 Taktperioden, was bei 8 MHz einer Zeit von 2,125 µs entspricht.

In die Überlegungen muß auch die Buskopplung des Systems einbezogen werden, wie sie in *Bild 2* dargestellt ist. Die Schlußfolgerungen aus Tabelle 1 sind in *Tabelle 2* aufgeführt. Ihr liegt die einfache Befehlssequenz „Lesen-Schreiben" zugrunde. Es fällt auf, daß die Kombination LS-Treiber mit 200-ns-RAMs eine geringere Leistung zur Folge hat als die Lösung mit Schottky-Treibern und herabgesetzter Taktfrequenz. Einen erstaunlichen Leistungswert ergibt die letzte Zeile, die sich auf die Zusatzschaltung in *Bild 3* bezieht.

Wie aus *Bild 4* zu ersehen ist, wird sozusagen auf Wunsch des langsameren Peripheriebausteins der Prozessortakt S4 gedehnt, ohne daß es der Prozessor überhaupt merkt. Dadurch entfallen zusätzliche WAIT-Zyklen, und die Leistungseinbuße bleibt gering.

Dipl.-Ing. Pal Poth

Dynamisches RAM für 68000-Systeme

Dynamische 64-KBit-RAMs dürften bald von den 256-KBit-Typen überflügelt werden. Wir stellen Ihnen eine Speicherkarte für 68000-Systeme vor, die diesem Umstand Rechnung trägt. Je nach Bestückung hat sie eine Kapazität von 128 oder 512 KByte. Speziell programmierte PAL-Bausteine oder teure Refresh-Controller benötigt sie nicht.

Da beim 68000 kein Hardwarerefresh vorhanden ist, tut sich dieser Chip nicht so leicht mit dynamischen Speichern wie z. B. der Z80. Softwarerefresh, z. B. über die Interruptprioritätsebene 7, oder gar Abschalten der CPU über die Buslogik behindern den Betrieb des Systems doch erheblich. Da der Prozessor jedoch nicht unentwegt auf den Speicher zugreift, kann der Refreshzyklus auch in der Phase stattfinden, in der die CPU keinen Speicherzugriff vornimmt. Der 68000 ist ein asynchroner Prozessor. Nach Anlegen von Daten und Adressen wartet er auf das DTACK-Signal des angesprochenen Elements und beendet danach den Zugriff.

Funktion der Schaltung

Adressiert wird der Speicher von den Signalen A1...A16 (A1...A18 bei der 512-KByte-Version) sowie durch UDS bzw. LDS, die ja beim 68000 die Leitung A0 ersetzen. RAMSEL aktiviert die Karte. Das Timing der dynamischen RAMs erfolgt quarzgesteuert mit Schieberegistern. Der 20-MHz-Oszillator, bestehend aus IC25a/b, erzeugt ein symmetrisches Rechtecksignal von 50 ns Zykluszeit, das den ganzen Ablauf taktet.

Zum Verständnis der Arbeitsweise sei vorausgesetzt, daß zum betrachteten Zeitpunkt kein Refreshvorgang abläuft. R/W, Daten und Adressen stehen stabil an, RAMSEL wird aktiv (low).

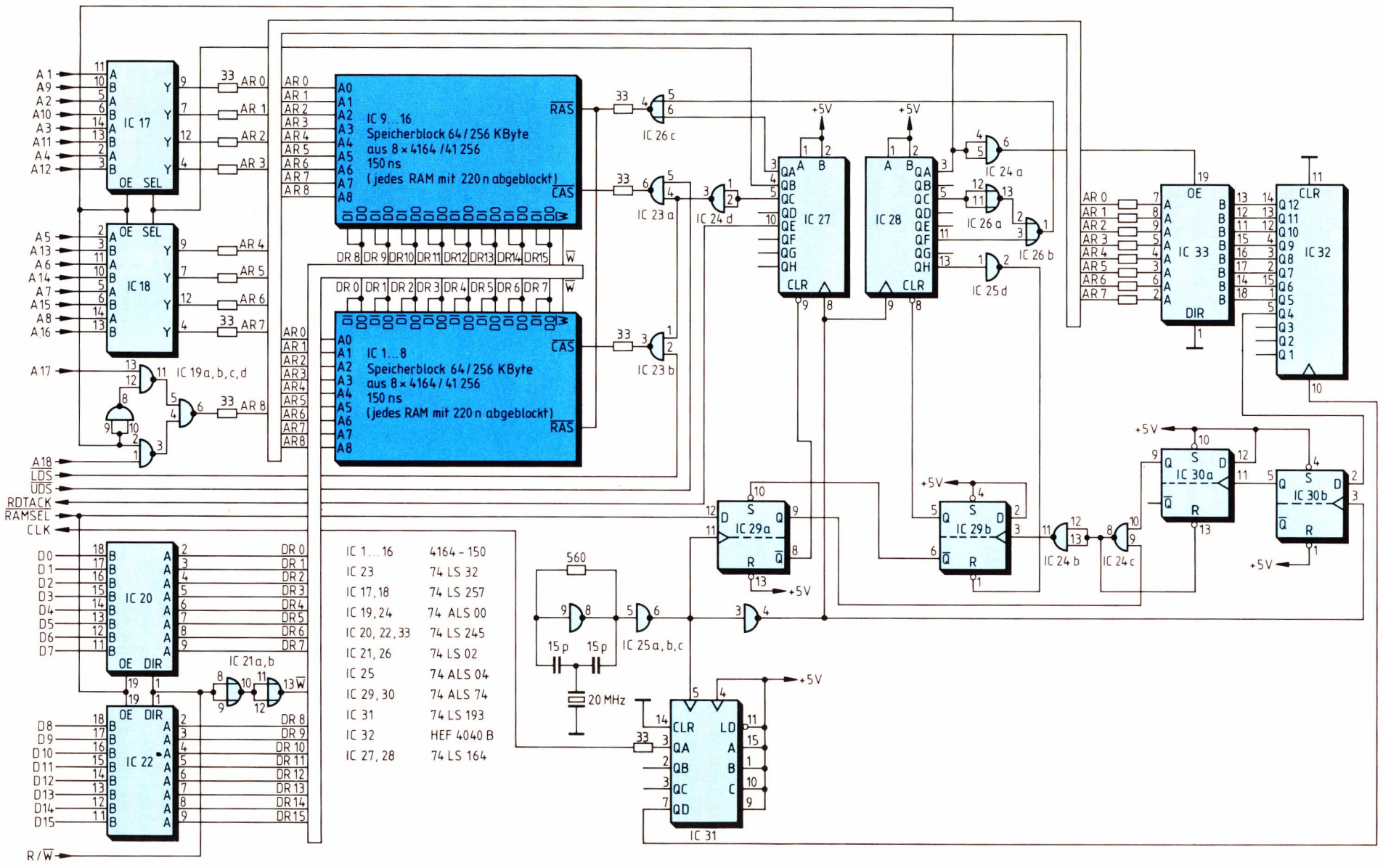
IC20 und 22, die bidirektionalen Datenpuffer, erhalten das Output-Enable-Signal. Mit positiver Flanke des 20-MHz-Taktes übernimmt IC29a RAMSEL, sperrt über IC24c die Refreshlogik und gibt gleichzeitig Schieberegister IC27 frei. Damit beginnt im 50-ns-Takt die RAM-Ansteuerung: RAS – Multiplexer (IC17/18) umschalten – CAS – 100 ns Pause – DTACK ausgeben. DTACK ist hier positiv, da auf der Prozessorplatine durch NOR-Verknüpfung verschiedener DTACK-Signale das Prozessor-DTACK die richtige Polarität hat. Mit der Deselektierung der RAM-Karte wird IC27 wieder in den Clear-Zustand gesetzt, RAS und CAS werden zurückgenommen, wenn IC29a RAMSEL übernommen hat. Der Speicherzyklus ist beendet.

Jetzt kann der Refreshzyklus, falls nötig, stattfinden. Die Schaltung stellt eine 8-Bit-Refreshadresse zur Verfügung, die in 3,2 ms 256 Refreshadressen bzw. in 1,6 ms 128 Adressen zur Verfügung stellt. Es können also ICs mit 128/256er-Refresh sowie die 256-KBit-Typen benutzt werden. Die Zähler IC31/32 erzeugen durch Herunterteilen des Oszillatorsignals die Refreshadresse. Q5...Q12 von IC32 liefern die Adresse; Q4 ist sozusagen der Taktgenerator für den nachfolgenden, negativ getriggerten Zählerteil. Daraus folgt, daß bei positiver Flanke an Q4 der Zähler IC32 einen stabilen Ausgangszustand angenommen hat, den er dem Tristate-Puffer IC33 über Q5...Q12 übergibt. IC30b übernimmt Q4 von IC32 mit der negativen Flanke des 20-MHz-Taktes. Sobald Q4 den Pegel gewechselt und die Übernahme stattgefunden hat, wird IC30a zur Übernahme von 1 zum Ausgang veranlaßt und sofort über IC24c zurückgesetzt. Der dabei entstehende Impuls, über IC24b an IC29b weitergeleitet, veranlaßt IC29b, das Schieberegister IC29 freizugeben und den regulären Speicherzugriff zu sperren. Als erstes schaltet IC29 die Multiplexer IC17/18 ab. 100 ns später wird für die Dauer von 150 ns RAS gegeben und die Refreshadresse über das gleichzeitig mit dem Abschalten von IC17/18 aktivierte IC33 übernommen. 100 ns nach dem Ende von RAS wird IC29b zurückgesetzt. Der Refreshzyklus von 400 ns Dauer ist vorbei.

Damit sich der Speicherzugriff der CPU und der Refreshzyklus nicht gegenseitig „auf die Füße treten", ist noch etwas Steuerlogik nötig.

Über IC29a wird RAMSEL mit der positiven Flanke des 20-MHz-Taktes übernommen, die Refreshanforderung mit der negativen Flanke über IC30b. Damit besteht unabhängig vom zeitlichen Zusammentreffen von RAMSEL und der Refreshanforderung ein minimaler Abstand von 25 ns, der ausreicht, um den nicht angeforderten Zyklus für die Dauer des laufenden sicher zu sperren. TTL-ICs der S- oder ALS-Serie erfüllen diese Anforderungen.

Läuft ein CPU-Speicherzugriff ab, hält IC29a über IC24c IC29b davon ab, einen Refreshzyklus zu starten. Läuft ein Refreshzyklus, hält IC29b das IC29a über den Setzeingang fest, so daß kein CPU-Speicherzugriff möglich ist. Die CPU bekommt somit auch kein DTACK und muß warten, bis sie an der Reihe ist. Bestünde das ganze abzuarbeitende Programm aus Speicherzugriffen, hätte diese Art des Refresh eine Geschwindigkeitsreduzierung von 3 % zur Folge. Im praktischen Betrieb bleibt die Einbuße jedoch unter 1 %. Eines muß jedoch beachtet werden: Durch Zurückhalten des DTACK-Signals wird gelegentlich per Hardware Einzelschrittbetrieb „gefahren". Das geht hier nicht, denn solange die CPU ihr DTACK nicht hat, bleibt das RAM adressiert, also RAMSEL auf 0. Dann kann aber kein Refreshzyklus gestartet werden, die Folgen sind klar. *Dipl.-Ing. Hubert Reelsen*



Bis zu 256 KByte RAM kann auf dieser Speicherkarte untergebracht werden

Kassettenrecorder zeichnet Signale sehr niedriger Frequenz auf

Dieser Beitrag zeigt, wie man mit Hilfe eines Mikrocomputers, eines A/D-Umsetzers und minimaler Hardware sehr einfach und preisgünstig (für unter 100 DM) aus einem billigem Kassettenrecorder einen Recorder für die Aufnahme und Wiedergabe von Signalen im Frequenzbereich 0...100 Hz mit sehr niedriger Fehlerrate machen kann. Anwendungen für das Verfahren finden sich in vielen industriellen Bereichen.

Für das Speichern von sehr niederfrequenten analogen Signalen (wie zum Beispiel EKG-, EEG- oder seismische Signale) auf magnetischem Träger führt zwangsläufig zu einer frequenzmodulierten Aufzeichnung. Da Geräte mit FM-Aufzeichnung systembedingt hohe Ansprüche auf Gleichlauf und Bandgeschwindigkeit stellen und außerdem eine komplizierte analoge Elektronik beinhalten, sind sie sehr teuer und für die meisten Anwender leider unerschwinglich. Liegt die maximal aufzuzeichnende Frequenz aber unter 100 Hz, ist eine Digital-Aufzeichnung mit einer Bit-Rate von 2400 Baud durchaus möglich. Dann ist für die Aufnahme lediglich eine A/D-Umsetzung mit anschließender Parallel-Seriell-Umwandlung, und für die Wiedergabe eine Seriell-Parallel-Wandlung mit folgender D/A-Umsetzung ausreichend.

Die gesamte Elektronik und deren Steuerung vereinfacht sich, wenn ein Mikrocomputer in das System mit einbezogen wird. Die Blockschaltung (*Bild 1*) zeigt die einzelnen Teile des Systems. Vom Mikrocomputer wird die Steuerung des umschaltbaren A/D-/D/A-Umsetzers, die Seriell-Parallel-Umsetzung während der Wiedergabe, die Parallel-Seriell-Wandlung und die Biphase-Modulation (Manchester-Code) für die Aufnahme sowie die Überwachung des Eingangspegels während der Aufnahme übernommen. Beim A/D/D/A-Umsetzer handelt es sich um einen 8-Bit-Typ nach dem Wäge- oder Iterativverfahren (Sukzessive Approximation). Ein Tiefpaß dritter Ordnung mit einer Übergangsfrequenz von 100 Hz und eine Sample-and-Hold-Schaltung sorgen für die Signalverarbeitung. Den Umsetzer taktet der Mikrocomputer.

Im Wiedergabe-Teil befinden sich Signal-Decodierer und Taktaufbereiter. Dieser Block entzerrt und begrenzt das Wiedergabesignal, demoduliert das Biphase-Signal und gewinnt auch die serielle Taktfrequenz mittels eines PLL-Synthesizers zurück. Somit werden auch starke Bandgeschwindigkeits-Änderungen ausgeglichen.

Der Mikrocomputer *Bild 2* ist eine Minimalkonfiguration. Als CPU ist der Typ MC6802 mit internem Taktgenerator sowie 128 Byte RAM vorgesehen. In einem 2716

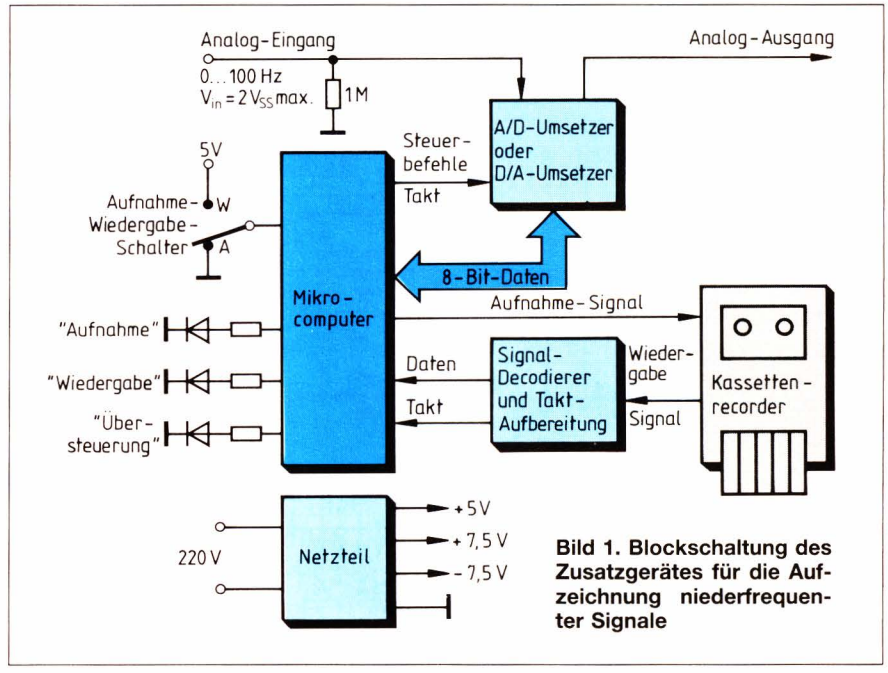


Bild 1. Blockschaltung des Zusatzgerätes für die Aufzeichnung niederfrequenter Signale

befindet sich die Software und für die periphere Ein-/Ausgabe sorgt ein PIA MC6821. Das NAND-Gatter 74LS00 sorgt für die sehr einfache Adressen-Decodierung. Vom EPROM wird die gesamte obere Hälfte des Adreßbereiches ($8000...$FFFF) belegt, der PIA belegt fast die ganze untere Hälfte ($0080...$7FFF) des Adreßbereichs. Zwischen $0000...$007F befindet sich der RAM-Bereich von 128 Byte in der CPU 6802, der als Stack dient. Der PIA wurde absichtlich auf die Adressen von $0080...$0083 decodiert, damit eine 8-Bit-Adressierung („zero-page-addressing") möglich ist. Dieses Vier-Chip-System stellt die preiswerteste und günstigste Lösung dar.

Das analoge Eingangssignal wird zuerst von einem FET-Operationsverstärker gepuffert und anschließend über einen Analogschalter (4053a) zu einem aktivem Tiefpaß dritter Ordnung geleitet (*Bild 3*). Um Phasenfehler auf ein Minimum zu beschränken, ist ein Butterworth-Filter eingebaut. Da die Aufzeichnungsrate auf Band 2400 Baud beträgt und neben den acht Datenbits ein Start- und zwei Stop-Bits verwendet wurden, ist die Abtastfrequenz mit 2400/11 = 218 Hz gegeben. Deshalb liegt auch die Übergangsfrequenz bei 100 Hz. Der Ausgang des Filters liegt wieder über einen analogen Schalter (4053b) an der Sample-and-Hold-Schaltung. Diese besteht aus einem Abtaster (4053d), dem Halte-Kondensator C und nachfolgendem FET-Operationsverstärker.

Der A/D-Umsetzer besteht aus mehreren Chips, weil er auch als D/A-Umsetzer funktionieren soll. Solange der 8-Bit-Puffer (74LS244) transparent ist, sind die acht Ausgänge des Sukzessive-Approximations-Registers SAR (MC14549) an den D/A-Umsetzer (1408) und an den PIA angeschlossen. Der D/A-Umsetzer hat einen Stromausgang und der folgende Operationsverstärker sorgt für die Strom-Spannungs-Umsetzung. Ein Komparator (311) vergleicht die momentane D/A-Ausgangsspan-

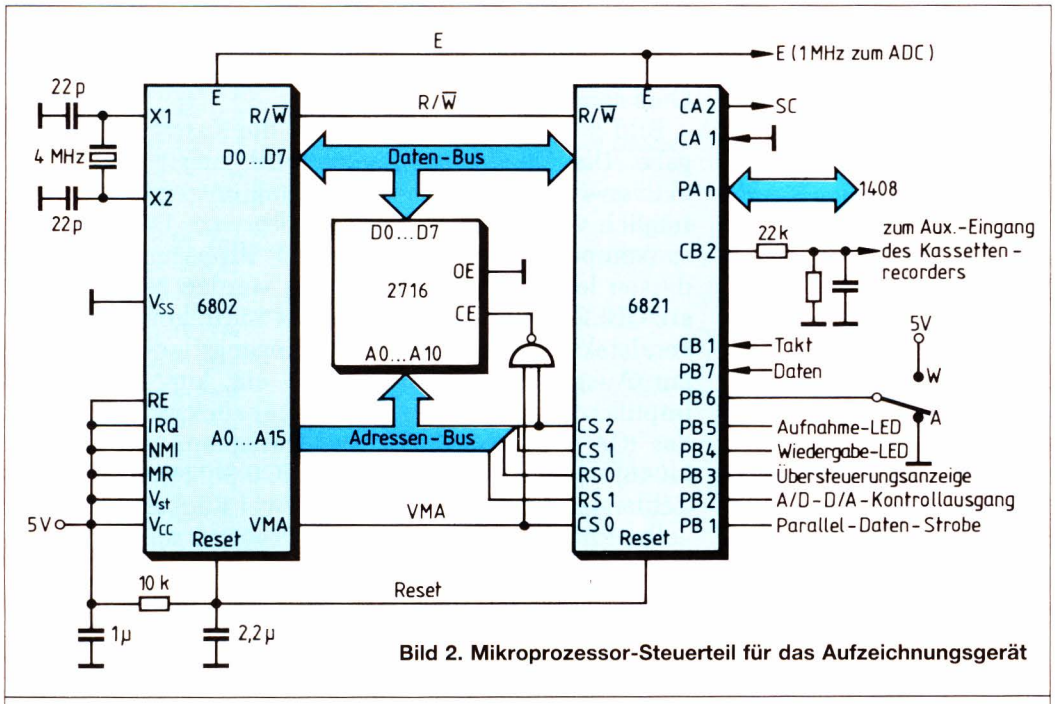


Bild 2. Mikroprozessor-Steuerteil für das Aufzeichnungsgerät

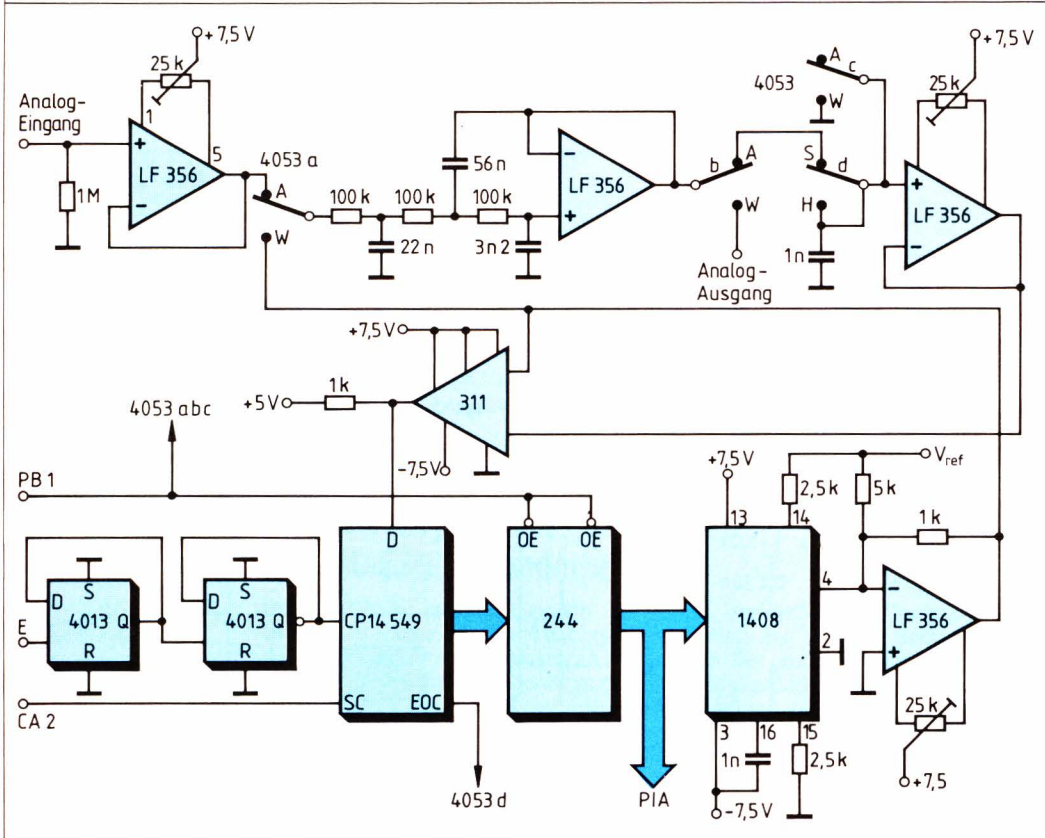


Bild 3. Aufnahmekanal des Aufzeichnungsgerätes

nung und die abgetastete Eingangsspannung. Der Komparatorausgang wird dem SAR zurückgeführt. Eine komplette A/D-Umsetzung erfolgt in neun Taktimpulsen. Da die A/D-Schleifenverzögerung ungefähr gleich 2 µs ist, muß die Taktfrequenz kleiner als 0,5 MHz sein. Deshalb wird auch der CPU-Takt E mittels zwei Flipflops (4013) durch vier geteilt. Eine A/D-Umsetzung wird vom Mikrocomputer gestartet, indem der SC-Eingang des SAR eine positive Flanke vom CA2 des PIA zugeschickt bekommt. Der EOC-Ausgang des SAR steuert den Abtaster (4053 d) der Sample-and-Hold-Schaltung. Während der fortschreitenden Umsetzung steht der Abtaster in Hold-Stellung (H), andererseits in Sample-Stellung (S). Die dynamischen Innenwiderstände der Analogschalter 4053 b und d sorgen für die Entkopplung der kapazitiven Last, verursacht durch den Halte-Kondensator. Die A-Seite des PIA-Bausteins ist während der Aufnahme als Eingang programmiert.

Bei Wiedergabe stehen die drei Analogschalter 4053 a, b und c in Stellung W und der Ausgang des D/A-Umsetzers führt über das Tiefpaßfilter zum Analog-Ausgang. Der Eingang der Sample-and-Hold-Schaltung liegt auf Massepotential, der Bustreiber 74LS244 ist im Tristate-Zustand, und die A-Seite des PIA funktioniert als Ausgang.

Wie schon vorher erwähnt, sorgt die CPU für die Parallel-Seriell-Umsetzung und umgekehrt. Da aber ein NRZ-Signal (Non Return to Zero) eine Gleichstromkomponente aufweist, ist es für die magnetische Aufzeichnung untauglich. Daher muß man das aufzuzeichnende Signal einer Modulation bzw. einer Codierung unterziehen. Biphase-Modulation (*Bild 4*) wurde gewählt, da es keine Gleichstrom- und auch keine sehr niederfrequenten Komponenten aufweist. Außerdem beinhaltet das Signal auch den seriellen Takt, welcher bei der Wiedergabe leicht zurückzugewinnen ist. Ein weiterer Vorteil der Biphase-Modulation ist, daß eine Bandbreite des Kassettenrecorders von 4 kHz durchaus ausreichend ist, vorausgesetzt, daß der Phasengang konstant ist. Die geringe Bandbreite ist von großem Vorteil, da Kopfverschleiß und Kopfspalt-Justierfehler weniger ins Gewicht treten, und auch keine besonderen Anforderungen an die Qualität der Bänder bestehen. Die Kassetten müssen lediglich mechanisch einwandfrei und mit wenig Drop-Outs sein.

Bild 5 zeigt die Schaltung für die Kassetten-Wiedergabe. Das Wiedergabe-Signal wird zuerst von einem aktiven Filter (741) im Phasengang entzerrt und nachträglich vom Komparator (311) begrenzt. Das nun phasenkompensierte und begrenzte Biphase-Signal und dessen leicht verzögerte Version werden in ein Exklusiv-ODER-Gatter geführt, das als bidirektionaler Flankendetektor arbeitet. Bei jeder Eingangsflanke erscheint am Ausgang von IC3a (4070) ein kurzer positiver Impuls, der den Zähler IC4 (4024) rücksetzt. Der Takt des IC4 wird von der Ausgangsfrequenz des PLL-IC5 (4046) nach Teilen durch 2 von IC6 (4024) erzeugt. Der Zählerstand von IC4 zwischen zwei Rücksetzimpulsen sollte für H-Pegel 32 und für L-Pegel 16 sein. Am Ausgang von IC7a liegt nach 24 Impulsen L-Pegel, also halbwegs zwischen den beiden Extremen. IC8a, b und IC3b resynchronisieren und symmetrieren den Ausgangsdatenstrom. Bei korrekter Bandgeschwindigkeit beträgt die Ausgangsfrequenz der PLL 153,6 kHz. Am Ausgang von IC3c liegt die serielle Taktfrequenz von 2400 Hz, die am flankenempfindlichen Eingang CB1 des

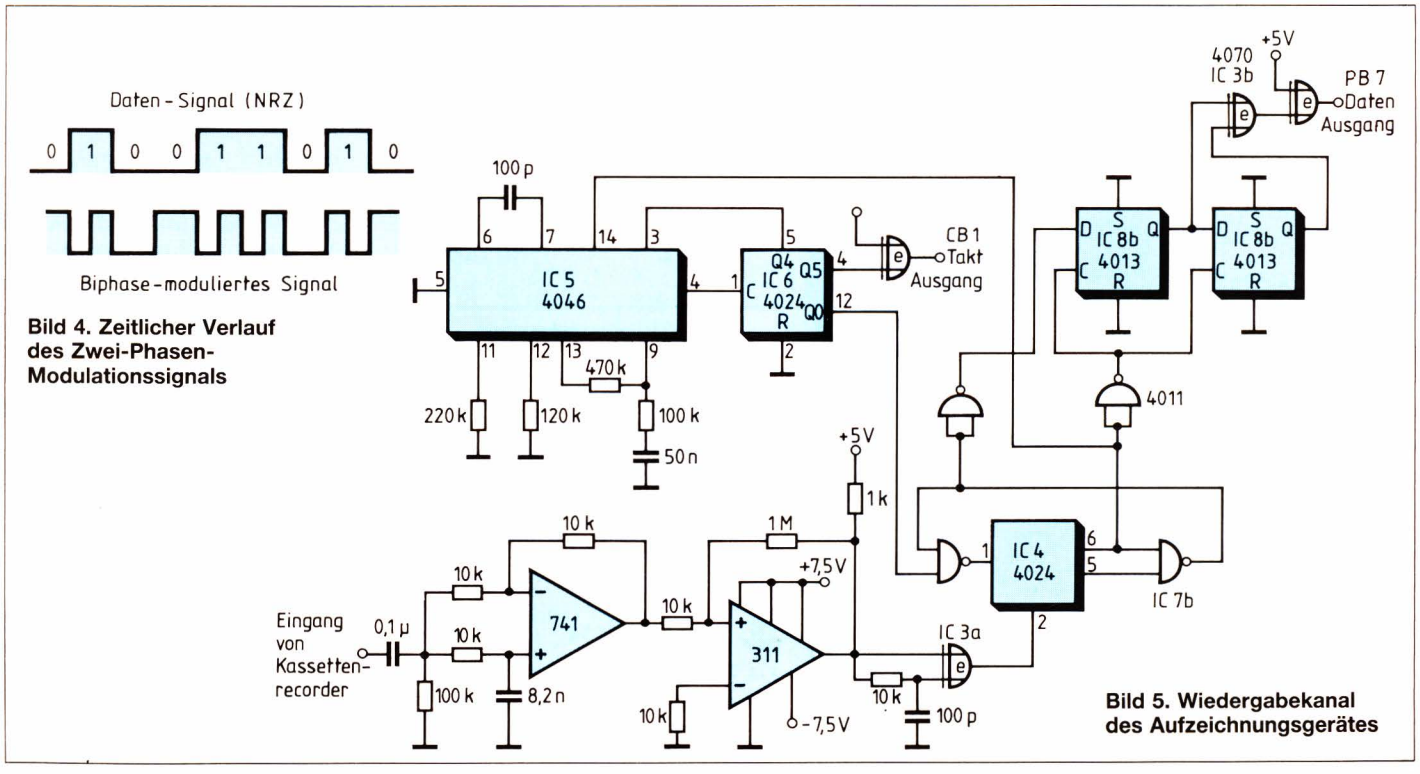


Bild 4. Zeitlicher Verlauf des Zwei-Phasen-Modulationssignals

Bild 5. Wiedergabekanal des Aufzeichnungsgerätes

```
00001  *
00002  *        PROGRAM DES ANALOG-DATEN
00003  *                RECORDERS
00004  *
00005  *
00006                          NAM    INSTAP
00007                          OPT    REL
00008  *
00009          0080  A PIA     EQU    $80       PIA BASIS ADDRESSE
00010  *
00011A FC00                    ORG    $FC00     PROGRAMM FAENGT HIER AN
00012  *
00013A FC00 8E 007F  A RESET   LDS    #$7F      STACK
00014A FC03 C6 04    A         LDAB   #4
00015A FC05 D7 83    A         STAB   PIA+3
00016A FC07 86 40    A         LDAA   #$40      ? AUFNAHME
00017A FC09 95 82    A         BITA   PIA+2
00018A FC0B 27 47 FC54         BEQ    RECORD    AUFNAHME
00019  *
00020A FC0D 7F 0083  A PLAYBK  CLR    PIA+3     WIEDERGABE
00021A FC10 86 17    A         LDAA   #$17      PIA KONFIGURATION
00022A FC12 97 82    A         STAA   PIA+2     FUER WIEDERGABE
00023A FC14 D7 83    A         STAB   PIA+3
00024A FC16 86 12    A         LDAA   #$12      D/A MODUS
00025A FC1B 97 82    A         STAA   PIA+2     WIEDERGABE LED LEUCHTET
00026A FC1A 4F          NXSAMP CLRA             DATA VOM TAPE
00027A FC1B C6 08    A         LDAB   #8
00028A FC1D 8D 27 FC46 NXBIT   BSR    CHKBIT
00029A FC1F 49                 ROLA
00030A FC20 5A                 DECB
00031A FC21 26 FA FC1D         BNE    NXBIT
00032A FC23 8D 21 FC46 SPST1   BSR    CHKBIT    TEST FUER START/STOP
00033A FC25 24 FC FC23         BCC    SPST1     BITS
00034A FC27 8D 1D FC46         BSR    CHKBIT
00035A FC29 24 F8 FC23         BCC    SPST1
00036A FC2B 8D 19 FC46 SPST2   BSR    CHKBIT
00037A FC2D 25 FC FC2B         BCS    SPST2
00038A FC2F 4D                 TSTA
00039A FC30 2A 04 FC36         BPL    BRAPL
00040A FC32 84 7F    A         ANDA   #$7F
00041A FC34 20 02 FC38         BRA    OUTSIG
00042  *
00043A FC36 8A 80    A BRAPL   ORAA   #$80
00044A FC38 97 80    A OUTSIG  STAA   PIA
00045A FC3A D6 82    A         LDAB   PIA+2     SENDE DATA GUELTIG
00046A FC3C CA 04    A         ORAB   #$04      IMPULS
00047A FC3E D7 82    A         STAB   PIA+2
00048A FC40 C4 FB    A         ANDB   #$FB
00049A FC42 D7 82    A         STAB   PIA+2
00050A FC44 20 D4 FC1A         BRA    NXSAMP
00051  *
00052A FC46 7D 0083  A CHKBIT  TST    PIA+3
00053A FC49 2A FB FC46         BPL    CHKBIT
00054A FC4B 7D 0082  A         TST    PIA+2
00055A FC4E 2A 02 FC52         BPL    BIT0
00056A FC50 0D                 SEC
00057A FC51    85    A         FCB    $85       SKIP 1
00058A FC52 0C          BIT0   CLC
00059A FC53 39                 RTS
00060  *
00061A FC54 7F 0083  A RECORD  CLR    PIA+3     AUFNAHME
00062A FC57 86 2F    A         LDAA   #$2F      PIA KONFIGURATION
00063A FC59 97 82    A         STAA   PIA+2     FUER AUFNAHME
00064A FC5B D7 83    A         STAB   PIA+3
00065A FC5D 86 29    A         LDAA   #$29      A/D MODUS
00066A FC5F 97 82    A         STAA   PIA+2     AUFNAHME LED LEUCHTET
00067A FC61 4F          SATLP2 CLRA
00068A FC62 36          SATLP1 PSHA
00069A FC63 C6 3E    A         LDAB   #$3E      KONTROLLE FUER
00070A FC65 D7 81    A         STAB   PIA+1     UEBERSTEUERUNG
00071A FC67 C6 36    A         LDAB   #$36
00072A FC69 D7 81    A         STAB   PIA+1
00073A FC6B 8D 46 FCB3         BSR    XMBYTE
00074A FC6D 96 80    A         LDAA   PIA
00075A FC6F 81 00    A         CMPA   #$00
00076A FC71 27 04 FC77         BEQ    SATLOP
00077A FC73 81 FF    A         CMPA   #$FF
00078A FC75 26 03 FC7A         BNE    SATLP3
00079A FC77 32          SATLOP PULA
00080A FC78 20 E7 FC61         BRA    SATLP2
00081A FC7A 32          SATLP3 PULA
00082A FC7B 4C                 INCA
00083A FC7C 26 E4 FC62         BNE    SATLP1
00084A FC7E 7F 0083  A         CLR    PIA+3     MEHR VORBEREITUNGEN
00085A FC81 C6 04    A         LDAB   #4        FUER AUFNAHME
00086A FC83 86 2F    A         LDAA   #$2F
00087A FC85 97 82    A         STAA   PIA+2
00088A FC87 D7 83    A         STAB   PIA+3
00089A FC89 86 21    A         LDAA   #$21
00090A FC8B 97 82    A         STAA   PIA+2
00091A FC8D C6 3E    A RCLOOP  LDAB   #$3E      START IMPULS FUER A/D
00092A FC8F D7 81    A         STAB   PIA+1     UMSETZER (SC)
00093A FC91 C6 36    A         LDAB   #$36
00094A FC93 D7 81    A         STAB   PIA+1
00095A FC95 D6 82    A         LDAB   PIA+2
00096A FC97 8D 1A FCB3         BSR    XMBYTE    SENDE EIN BYTE (AUFNAHME)
00097A FC99 96 80    A         LDAA   PIA
00098A FC9B D6 82    A         LDAB   PIA+2
00099A FC9D CA 04    A         ORAB   #$04
00100A FC9F D7 82    A         STAB   PIA+2
00101A FCA1 C4 F3    A         ANDB   #$F3
00102A FCA3 D7 82    A         STAB   PIA+2
00103A FCA5 81 FF    A         CMPA   #$FF      ? UEBERSTEUERUNG
00104A FCA7 27 04 FCAD         BEQ    SATUR
00105A FCA9 81 00    A         CMPA   #$00
00106A FCAB 26 04 FCB1         BNE    LOPBAK
00107A FCAD CA 08    A SATUR   ORAB   #$08
00108A FCAF D7 82    A         STAB   PIA+2
00109A FCB1 20 DA FC8D LOPBAK  BRA    RCLOOP
00110  *
00111  *  DIESES PROGRAM SENDET EIN BYTE IN BIPHASE MODULATION
00112  *
00113A FCB3 8D 14 FCC9 XMBYTE  BSR    XMIT0     START BIT
00114A FCB5 C6 08    A         LDAB   #8
00115A FCB7 37          NEXBIT PSHB              NAECHSTES BIT
00116A FCB8 48                 ASLA
00117A FCB9 25 03 FCBE         BCS    TRMIT1
00118A FCBB 8D 0C FCC9         BSR    XMIT0
00119A FCBD    8C    A         FCB    $8C
00120A FCBE 8D 0E FCCE TRMIT1  BSR    XMIT1
00121A FCC0 33                 PULB
00122A FCC1 5A                 DECB
00123A FCC2 26 F3 FCB7         BNE    NEXBIT
00124A FCC4 8D 08 FCCE         BSR    XMIT1     1. STOP BIT
00125A FCC6 8D 06 FCCE         BSR    XMIT1     2. STOP BIT
00126A FCC8 39                 RTS
00127  *
00128A FCC9 8D 08 FCD3 XMIT0   BSR    INVERT    SENDE NULL
00129A FCCB 8D 13 FCE0         BSR    DELAY
00130A FCCD 39                 RTS
00131  *
00132A FCCE 8D 03 FCD3 XMIT1   BSR    INVERT    SENDE EINS
00133A FCD0 8D 01 FCD3         BSR    INVERT
00134A FCD2 39                 RTS
00135  *
00136A FCD3 C6 08    A INVERT  LDAB   #8
00137A FCD5 D5 83    A         BITB   PIA+3
00138A FCD7 27 03 FCDC         BEQ    PULHIG
00139A FCD9 C6 30    A PULLOW  LDAB   #$30
00140A FCDB    8C    A         FCB    $8C       SKIP 2
00141A FCDC C6 38    A PULHIG  LDAB   #$38
00142A FCDE D7 83    A         STAB   PIA+3
00143  *
00144A FCE0 C6 19    A DELAY   LDAB   #$19      VERZOEGERUNG
00145A FCE2 5A          DYLOOP DECB
00146A FCE3 26 FD FCE2         BNE    DYLOOP
00147A FCE5 39                 RTS
00148  *
00149A FFFE                    ORG    $FFFE
00150A FFFE    FC00  A         FDB    RESET
00151                          END
TOTAL ERRORS 00000--
```

Bild 6. Steuerprogramm für den Mikroprozessor

PIA-Bausteins anliegt.

Bei Aufnahme übernimmt der Mikrocomputer die Biphase-Modulation und am Ausgang CB2 des PIA-Bausteins steht das modulierte Signal zur Verfügung, das nach einer Tiefpaß-Filterung (6,5 kHz) zum Aux.-Eingang des Kassettenrecorders geführt wird.

Das System wurde mit verschiedenen Kassettenrecordern und Bändern getestet, wobei sich sehr zuverlässige und fehlerarme Ergebnisse zeigten.

Dr.-Ing. Ömer Cerid

Das MC-68000-Sonderheft

Gut eineinhalb Jahre gibt es ihn nun schon, den MC-68000-Computer, und mehrere tausend mc-Leser haben ihn nachgebaut. Höchste Zeit also, alle Beiträge, die in „mc" zu diesem erfolgreichen Bauvorschlag im Laufe der Zeit erschienen sind, in einem Sonderheft zusammenzufassen. Darüber hinaus enthält das Sonderheft einige bisher nicht veröffentlichte Beiträge, wie etwa einen IEC-Bus-Anschluß mit VIA-Bausteinen. Der Besitzer eines MC-68000 findet in diesem Sonderheft eine Zusammenfassung aller Informationen. Der an der Hardware Interessierte erhält einen Einblick in ein interessantes und überzeugendes Rechnerkonzept. Eine Vielzahl von Beispielen, von System- und Dienstprogrammen bis hin zur Einbindung von Grafik-Routinen in der Sprache C sind für den gedacht, der Prozessoren der Familie 68000 programmiert. Das Sonderheft kostet 22 DM. Wenn Sie mehr darüber wissen wollen, wenden Sie sich bitte an Ulrich Kruppe, Redakteur der „mc", Karlstraße 41, 8000 München 2, Tel. (0 89) 51 17-2 17.

Verbesserte Einzelschrittschaltung für Z80

Die im folgenden vorgestellte Schaltung stellt eine Verbesserung des in [1] beschriebenen Konzeptes dar. Sie hält den Z80 an einer einstellbaren Adresse an und ermöglicht daraufhin den Einzelschrittbetrieb.

Wenn der Prozessor „steht", kann er schrittweise mit der Taste „Einzelschritt" (*Bild 1*) weiter durch das Programm geführt werden. Man kann aber auch eine neue Stoppadresse über die Hexadezimal-Schalter eingeben und den µC mit Hilfe der Run/Stop-Taste starten. Er arbeitet dann das Programm weiter ab und hält an der neu eingegebenen Stelle, wodurch man in der Lage ist, ganze Programmabschnitte zeitsparend zu testen.

Der Daten- und der Adreßbus sind mit Hilfe von drei SN 74LS241 (IC1...IC3) gepuffert (*Bild 2*). Die Puffer und die Logik befinden sich auf einer Europakarte, die direkt auf den Bus des Z80 gesteckt werden kann.

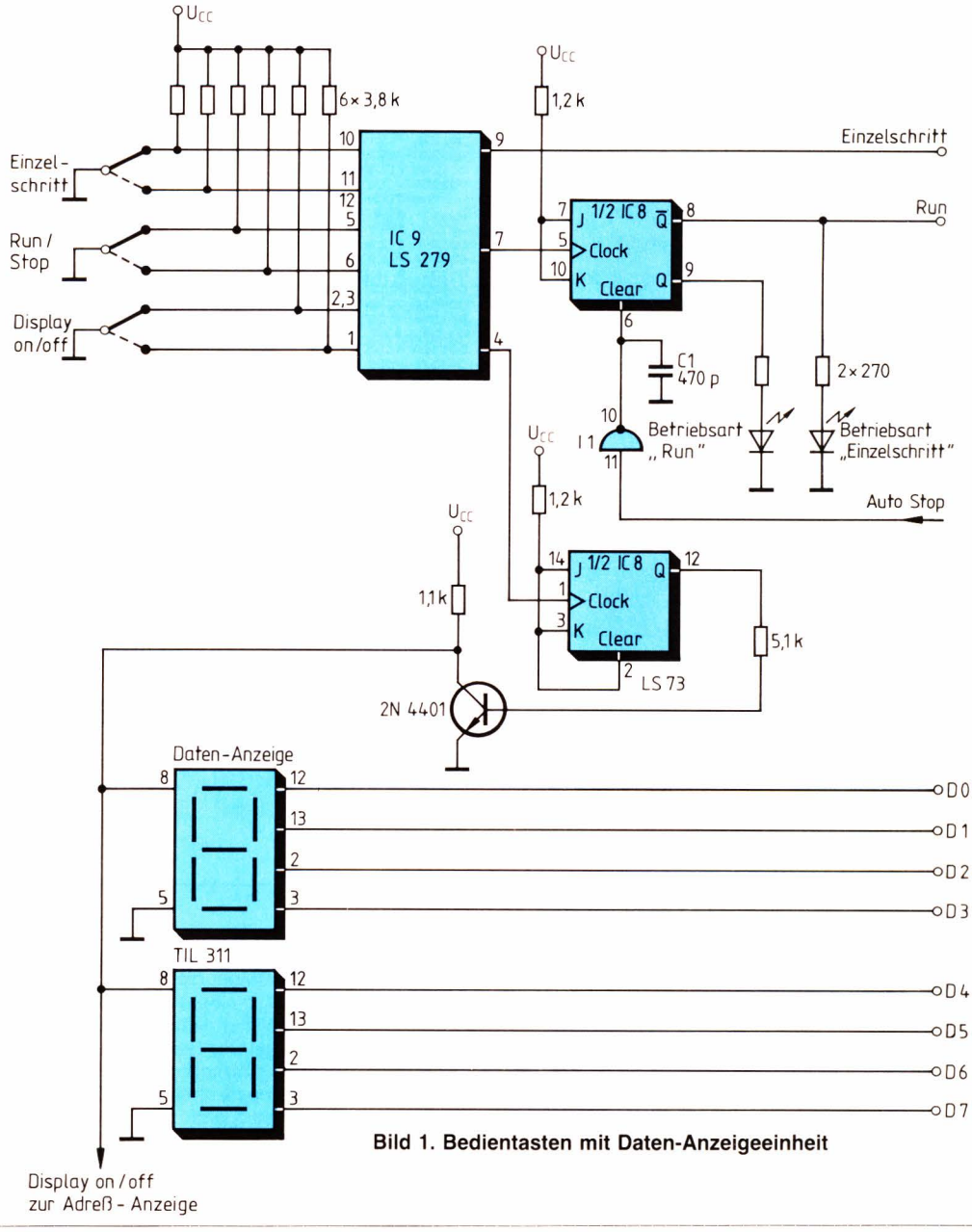


Bild 1. Bedientasten mit Daten-Anzeigeeinheit

Die gepufferten Daten- und Adreßleitungen gelangen über ein Flachbandkabel (24adrig) auf das Bedienungspult (*Bilder 1* und *3*). Sie führen dort direkt auf Hexadezimal-Displays (TIL 311). Die Adreßleitungen führen aber ebenso auf vier 4-Bit-Komparatoren (SN 74LS85). An die Komparatoren (IC4...IC7) sind über Invertierer außerdem Hexadezimal-Schalter angeschlossen. Stimmt die auf den Schaltern eingestellte Adresse mit der auf dem Datenbus befindlichen überein, so entsteht auf dem Auto-Stop-Ausgang der Komparatoren ein Signal, das über den Invertierer I1 auf den Umschalter führt. Der Ausgang des Flipflops (½ SN 74LS73) gelangt über das Flachbandkabel und über die Gatter G2, G3, G5 auf das WAIT-Signal des Prozessors und stoppt ihn. Durch Betätigen der Einzelschritt-Taste wird der Z80 kurzzeitig freigegeben und hält nach dem Abarbeiten eines Befehls erneut.

Durch Drücken der Run/Stop-Taste läuft der Z80 normal weiter. Befindet sich der Prozessor im Run-Mode, kann er durch Drücken der Run/Stop-Taste von Hand in den Single-Step-Mode gebracht werden.

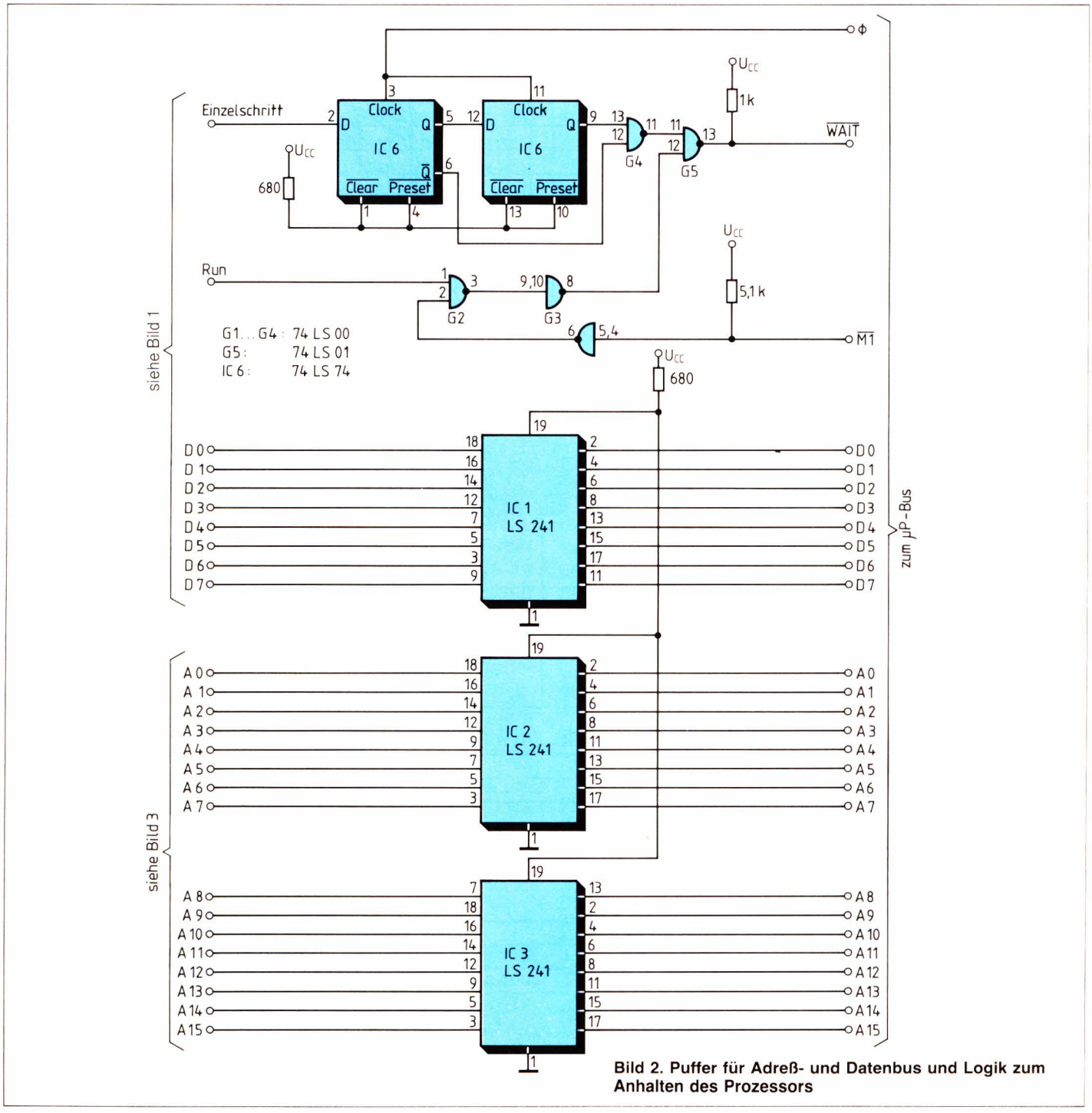


Bild 2. Puffer für Adreß- und Datenbus und Logik zum Anhalten des Prozessors

Mit der Taste „Display on/off" kann die gesamte Anzeigeeinheit abgeschaltet werden. Alle Tasten sind durch ein IC SN 74LS279 entprellt. Durch den Kondensator C1 wird die Betriebssicherheit erhöht, da er Störimpulse unterdrückt.

Hans Krause

Literatur

[1] *Riepe, J.*: Einzelschrittschaltung für den Mikroprozessor Z80. ELEKTRONIK 1979, H. 24, S. 100.

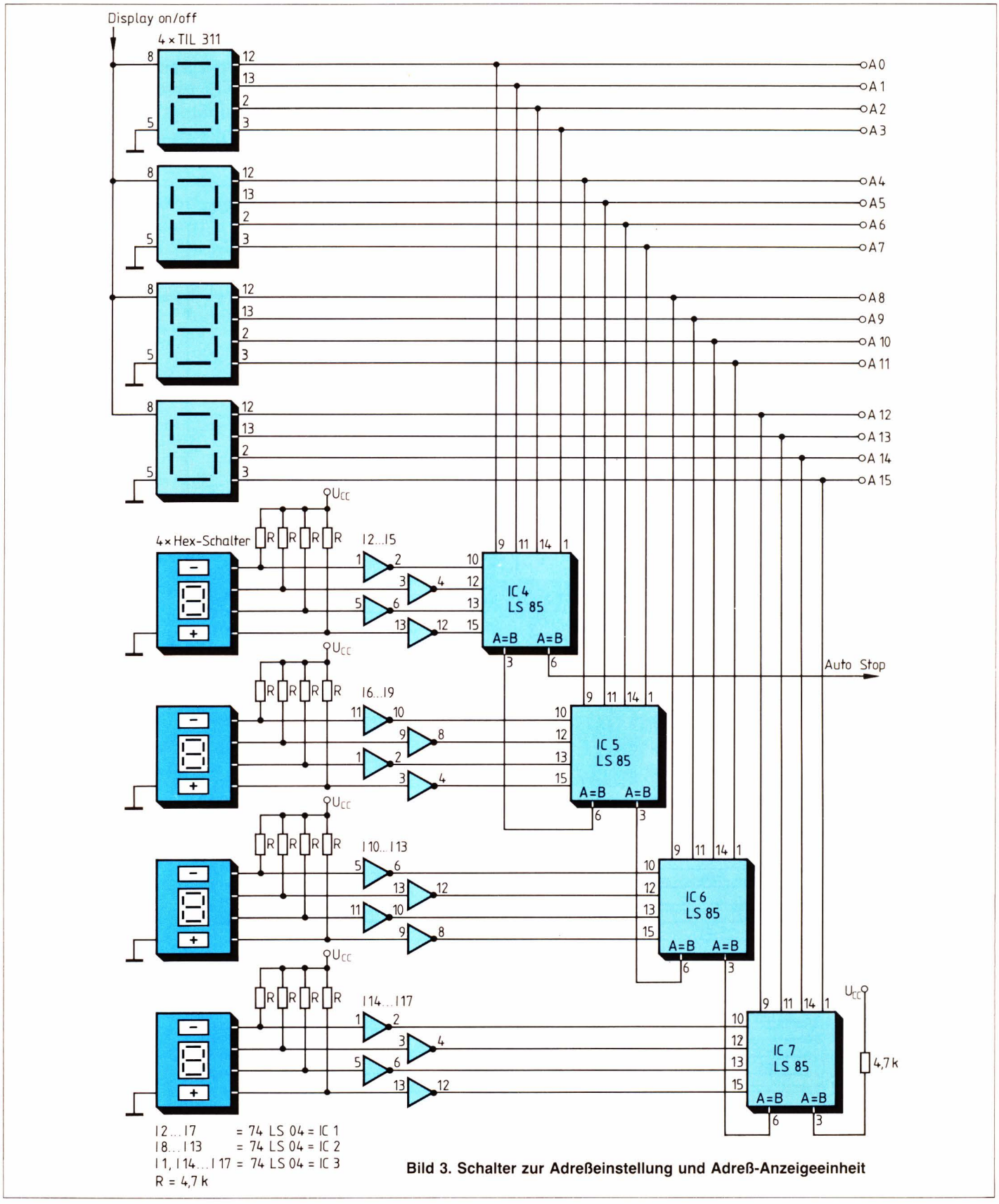


Bild 3. Schalter zur Adreßeinstellung und Adreß-Anzeigeeinheit

Einfaches Interface zwischen dynamischen RAMs und Z80-Systemen

Durch sogenanntes Bank-Selecting kann der adressierbare Speicherbereich von Z80-Systemen mit Hilfe des beschriebenen Interface erheblich erweitert werden. Ebenso nützlich ist in manchen Anwendungen die eingebaute Deselect-Möglichkeit.

Zwei Aufgaben muß das Interface erfüllen: Es muß die unteren (A0...A6) und oberen (A7...A13) sieben Adressen multiplexen und die zugehörigen Strobe-Signale ($\overline{RAS}$, $\overline{CAS}$) erzeugen; und es muß dafür sorgen, daß 128 Refresh-Zyklen in 2 ms ablaufen. Die zweite Forderung wird durch Verwendung des Z80 auf einfache Weise erfüllt. Dieser Prozessor enthält bereits den Refresh-Zähler, dessen Inhalt er nach jedem Opcode-Zyklus inkrementiert und auf den Bus legt. Ein spezielles Ausgangssignal des Z80 ($\overline{RFSH}$) zeigt diesen Zustand an. In der hier vorgestellten Schaltung wird von diesem Ausgangssignal aber kein Gebrauch gemacht. Der Refresh-Zyklus wird durch das $\overline{MRQ}$-Signal des Prozessors gestartet.

Durch den Einsatz eines PROMs konnte der Aufwand für das Interface reduziert werden, und zusätzliche Möglichkeiten wie Bank-Selecting und Deselecting (Ausblenden eines Adreßbereiches aus dem RAM für andere Aufgaben) wurden realisiert. Die in *Bild 1* dargestellte Schaltung erzeugt die Signale $\overline{RAS}$ und $\overline{CAS}$ durch Gatterlaufzeiten in den verwendeten ICs. Eine Schaltungsbeschreibung sei anhand des Funktionsablaufes eines Read-Zyklus und eines Refresh-Zyklus gegeben.

Read-Zyklus

Der Zyklus wird gestartet durch Aktivierung der $\overline{MRQ}$-Leitung des Prozessors. Zu diesem Zeitpunkt liegen bereits die Adressen auf dem Adreßbus. An den Ausgängen der Multiplexer IC 1 und IC 2 (74157, keine LS-Typen!) und damit an den sieben Adreßleitungen der RAMs liegen somit die Signale der Adressen A0...A6. Sie werden mit dem $\overline{RAS}$-Signal, das direkt dem über IC 5 gepufferten $\overline{MRQ}$-Signal entspricht, in die RAMs übernommen. Über Gatterlaufzeiten (je Gatter des 74LS365 etwa 16 ns) von IC 5 verzögert, werden anschließend die Multiplexer umgeschaltet und die Adressen A7...A13 an die Adreßeingänge der RAMs gelegt. Da es sich um einen Read- oder Write-Zyklus handelt, müssen diese Adressen ebenfalls in die RAMs übernommen werden. Das heißt es muß für den, entsprechend der Information der Adressen A14 und A15, selektierten 16-KByte-Block ein $\overline{CAS}$-Signal erzeugt werden. Decodierung und Generierung dieser Signale werden mit einem PROM 74S288 (32 × 8) auf einfache Weise durchgeführt. Das PROM ist so programmiert, daß nur dann ein $\overline{CAS}$-Signal erzeugt wird, wenn entweder die $\overline{READ}$- oder $\overline{WRITE}$-Leitung aktiv ist. Nur dann ist über Gatter 11, 12, 13 von IC 6 und einen freien Multiplexer von IC 1 der Eingang A des PROM logisch H. Ein Ausgang (Anschluß 5) des PROM wird verwendet, um zusammen mit dem $\overline{READ}$-Signal die beiden Datenpuffer IC 3 und IC 4 so zu steuern, daß die aus den RAMs gelesenen Daten auf den Systemdatenbus gelegt werden. Das geschieht aber nur dann, wenn die beiden Adreßeingänge D und E des PROM (Anschlüsse 13 und 14) logisch H sind. Das Gesagte gilt entsprechend auch für den Write-Zyklus.

Refresh-Zyklus

Der Refresh-Zyklus wird als sogenannter $\overline{RAS}$-only-Refresh [2] durchgeführt. Die Erzeugung des $\overline{RAS}$-Signales geschieht genau wie vorher beschrieben bei einem Read- oder Write-Zyklus. Da aber während der Refresh-Zyklen des Prozessors weder die $\overline{READ}$- noch die $\overline{WRITE}$-Leitung aktiviert ist, wird kein $\overline{CAS}$-Signal erzeugt, weil PROM-Eingang A (Anschluß 10) auf H bleibt. Ebenso unterbleibt eine Umschaltung der Datenpuffer IC 3 und IC 4.

Bank-Selecting

Hierunter versteht man die Erweiterung des Speicherbereiches von üblicherweise 64 KByte durch Mehrfachbelegung der gleichen Adressen des Prozes-

sors. Da aber normalerweise jeweils nur eine Bank aktiviert sein darf, werden die anderen ganz oder teilweise abgeschaltet. In kommerziellen Systemen (z. B. der Firma Cromemco) ist hierfür ein E/A-Port bei der Adresse 40H vorgesehen. Jedes der acht Bits dieses Ports kann eine Bank von maximal 64 KByte aktivieren. Es ist damit also möglich, einen Speicherbereich von 8 × 64 = 512 KByte in einem System zu adressieren. Hiervon wird z. B. in Mehrbenutzersystemen Gebrauch gemacht, wobei das Betriebssystem selbst in den oberen 16 KByte des 64-KByte-Adreßbereiches arbeitet und immer aktiviert bleibt. Jedem Benutzer stehen dann noch eine Bank mit bis zu 48 KByte zur Verfügung. Mit der hier vorgestellten Schaltung können auch kleinere Bereiche in Stufen zu 16 KByte gebildet und aktiviert werden, z. B. lassen sich damit auf einer Europakarte „Banks" mit je 32 KByte realisieren.

Deselecting

Eine weitere interessante Möglichkeit bietet die DESELECT-Leitung (Adreßeingang D, Anschluß 14 des PROM). Hat man ein Prozessorsystem mit

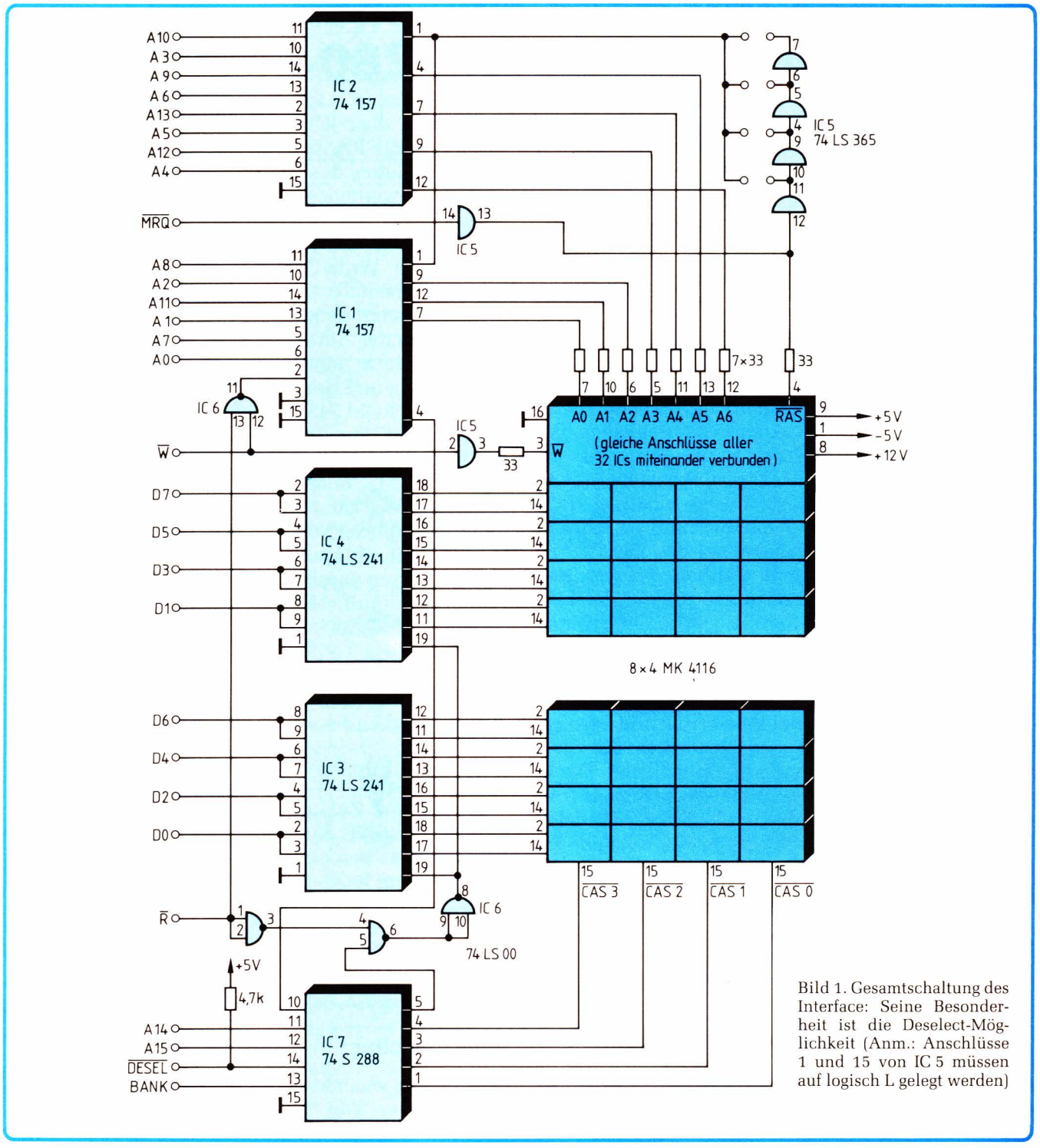


Bild 1. Gesamtschaltung des Interface: Seine Besonderheit ist die Deselect-Möglichkeit (Anm.: Anschlüsse 1 und 15 von IC 5 müssen auf logisch L gelegt werden)

Beispiel für die Programmierung des PROM 74S288 für 64 KByte in einer Bank mit Deselect-Möglichkeit

Adresse	Datum Bit 43210	Sed.	Adresse	Datum Bit 43210	Sed.
00	01111	EF	10	01111	EF
01	01111	EF	11	01111	EF
02	01111	EF	12	01111	EF
03	01111	EF	13	01111	EF
04	01111	EF	14	01111	EF
05	01111	EF	15	01111	EF
06	01111	EF	16	01111	EF
07	01111	EF	17	01111	EF
08	01111	EF	18	01111	EF
09	01111	EF	19	11110	FE
0A	01111	EF	1A	01111	EF
0B	01111	EF	1B	11101	FD
0C	01111	EF	1C	01111	EF
0D	01111	EF	1D	11011	FB
0E	01111	EF	1E	01111	EF
0F	01111	EF	1F	10111	F7

64 KByte RAM bestückt, dann bleibt im Adreßbereich der üblichen Mikroprozessoren kein Platz mehr für residente Programme in PROMs oder EPROMs.

Bei Prozessoren, die keinen vom Adreßbereich unabhängigen E/A-Bereich haben, muß auch hierfür im 64-KByte-Speicherbereich Platz vorgesehen werden (z. B. bei 6502, 6800, SC/MP und anderen). Üblicherweise werden alle benutzten Adreßbereiche aktiv selektiert. Der umgekehrte Weg, also aktive Deselektierung, ist in vielen Anwendungen effizienter. Es wird dabei davon ausgegangen, daß der gesamte 64-KByte-Bereich mit RAM belegt ist (was aber nicht erforderlich ist). Werden Adreßbereiche für andere Aufgaben benötigt, wird das RAM deselektiert. Schaltungsmäßig ist das auf einfache Weise zu realisieren (*Bild 2*). Ein EPROM wird z. B. wie üblich selektiert, und das Select-Signal wird über ein Gatter mit Open-Collector-Ausgang auf die $\overline{\text{DESELECT}}$-Leitung gelegt, die im Bussystem vorgesehen sein muß.

Programmierung des PROM

Aus *Bild 3* geht der Signalverlauf an den Adreßeingängen (5) und den verwendeten Datenausgängen (5) des PROM 74S288 hervor. Gezeigt sind die Abläufe bei einem Refresh-Zyklus, einem Read- oder Write-Zyklus und einem Read- oder Write-Zyklus mit deselektiertem RAM. Ein Beispiel für die Programmierung bei durchgehendem Adreßbereich von 64 KByte RAM in einer Bank und Deselect-Möglichkeit ist in der *Tabelle* angegeben. Anzumerken ist, daß das BANK-SELECT-Signal aktiv H ist und das DESELECT-Signal aktiv L.

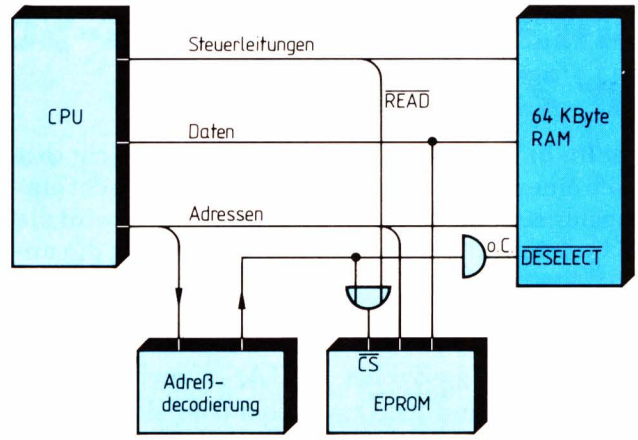


Bild 2. Hier wird mit der Deselect-Funktion ein Speicherbereich für ein EPROM ausgeblendet

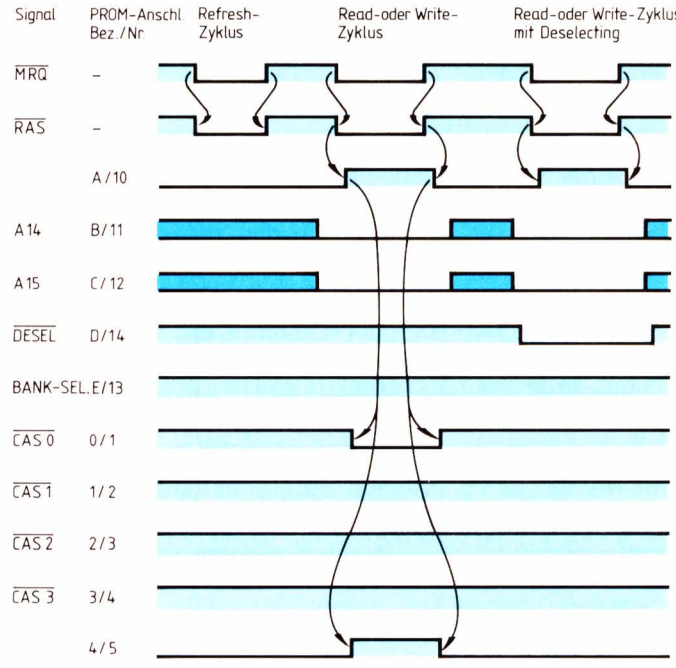


Bild 3. ▶
Signalverlauf an den Ein- und Ausgängen des PROM bei einem Refesh-Zyklus, einem Read- oder Write-Zyklus und einem Read- und Write-Zyklus mit deselektiertem RAM

Die gesamte Schaltung kann ohne Schwierigkeiten einschließlich der erforderlichen Stützkondensatoren für die drei Versorgungsspannungen auf einer Europakarte untergebracht werden.

Dr. H. B. Brinkhus, Wulfhard Schneider

Literatur

[1] Mikrocomputer Components Data Book 1979. Mostek.
[2] Memory Data Book & Designers Guide 1979. Mostek.

Einfache Interrupt-Schaltung für Z80-Systeme

Die im *Bild* dargestellte Schaltung ermöglicht den Betrieb eines Z80-Mikrocomputers mit bis zu acht einfach priorisierten Interrupt-Eingängen. Dabei wird die CPU im Interrupt-Modus 2 betrieben, bei dem die untere Hälfte des Interrupt-Vektors vom Datenbus, die obere durch das interne CPU-Register I geliefert wird. Die Daten vom Bus werden dabei während des Interrupt-Acknowledge-Zyklus (INTA) eingelesen, der durch $\overline{M1}$ = L und $\overline{IORQ}$ = L charakterisiert ist. Die Schaltung bildet für die Interrupt-Eingänge INT 0...7 die Vektoren F0, F2...FE (DB 0 ist immer L), wobei das obere Halbbyte F nicht aktiv von der Schaltung, sondern durch den „offenen" Datenbus erzeugt wird. Natürlich kann man auch vier zusätzliche Tristate-Puffer einsetzen, um DB 4...7 während der INTA-Zeit schalten zu können.

Da die Interrupt-Schaltung wegen des einfachen Konzepts den RETI-Befehl der CPU nicht erkennen kann, ist ihre Verwendung mit Z80-E/A-Bausteinen (CTC, PIO etc.) nur als Interrupt-Schaltung niedrigster Priorität möglich.

Das Funktionsprinzip ist einfach: Die bis zu acht Interrupt-Flipflops werden einzeln durch die positiv schaltenden Eingangsignale gesetzt. Dadurch ist gewährleistet, daß kein eintreffender Interrupt verlorengeht. Der nachfolgende Prioritäts-Codierer wählt den mit der höchsten Priorität eingeschalteten Interrupt aus und signalisiert ihn mit dem $\overline{INT}$-Signal der CPU. Ist diese bereit, den Interrupt anzuerkennen, und hat in der Zwischenzeit nicht noch eine höher priorisierte Schaltung einen Interrupt signalisiert (IEI wäre dann L), wird der höchste gespeicherte Interrupt mit INTA = H im Latch festgehalten und auf den Datenbus geschaltet. Gleichzeitig wird das Flipflop, dessen Interrupt quittiert wurde, wieder zurückgesetzt.

Je nach den Eigenschaften, die für die acht Interrupt-Eingänge gefordert werden, können anstelle der positiv flankengetriggerten D-Flipflops 74LS74 auch RS-Flipflops vom Typ 74LS279 benutzt werden. Dann sollte die Zeit, in der der INTn-Eingang aktiv (RS-Flipflop = L) ist, $3t_c$ des CPU-Taktes nicht überschreiten, da das Flipflop sonst nicht ordnungsgemäß gelöscht werden kann.

Der zusätzliche IEO-Ausgang, der anzeigt, daß diese Schaltung gerade einen Interrupt anfordert, ist mit den IEO-Ausgängen der Z80-Bausteine nicht identisch, da diese während des RETI-Befehls unter Umständen auf „H" geschaltet werden [3].

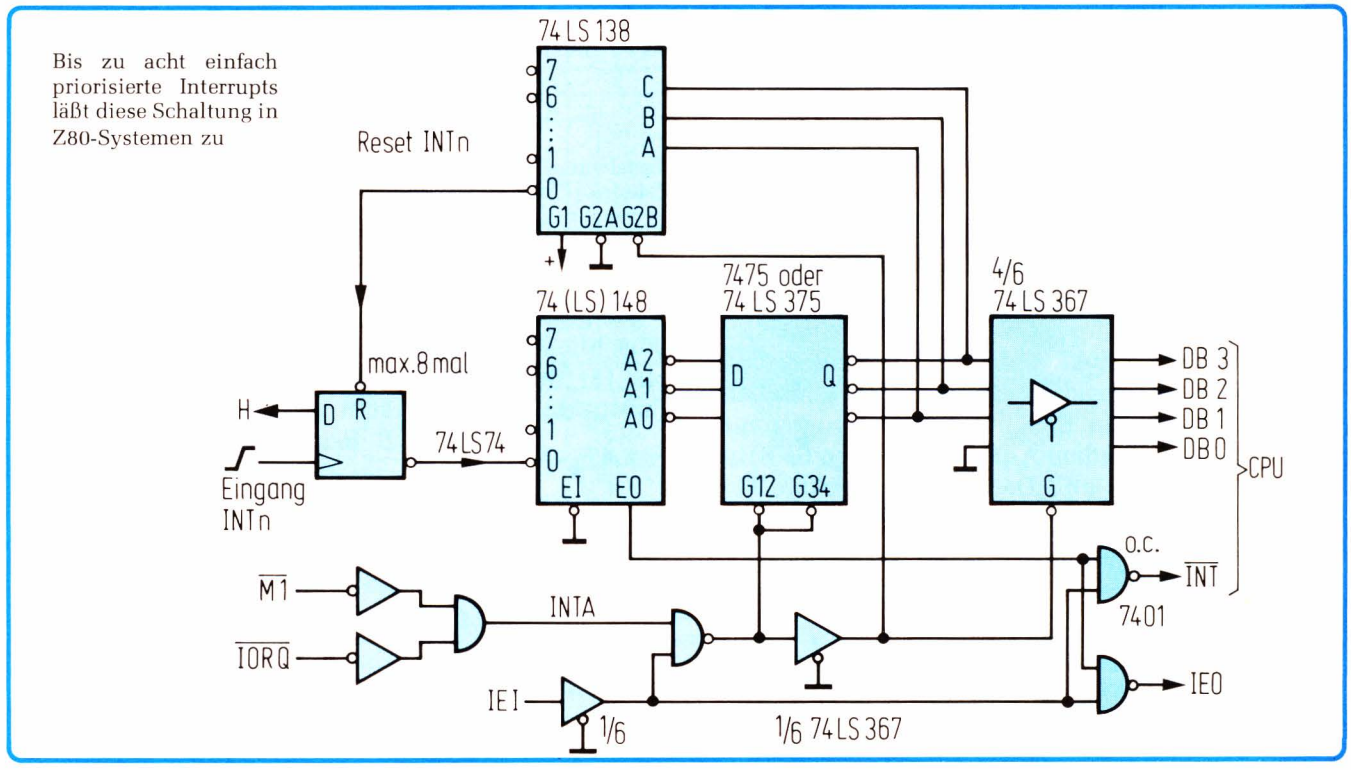
Thomas Gries

Literatur

1 Mikrocomputer 77/78 (Datenbuch). Zilog, 1977.

2 *Feger, O.*: Schnelle Schaltung zur Interrupt-Verarbeitung. ELEKTRONIK 1977, H. 11, S. 88.

3 *Prikryl, J.*: Interrupt-Behandlung im Z80-System und Konsequenzen für die E/A-Erweiterung. ELEKTRONIK 1980, H. 13, S. 55...59.



Schneller Multiplizierer für Mikrocomputer

Der Zeitbedarf für Multiplikationen ist bei MOS-Mikroprozessoren relativ groß. Zum Beispiel braucht der Prozessor 6502 für eine 32 × 32-Bit-Multiplikation mit 64 Bit Ergebnis je nach Anzahl der 1-Bits im Multiplikator zwischen etwa 1,7 und 9,5 ms [1]. Um diese Zeit zu verringern, ist vorgeschlagen worden, einen Parallelmultiplizierer, der bis zur Größe 24 × 24 Bit als monolithisch integrierte Schaltung erhältlich ist, anzuschließen [2]. Dies ist sicherlich eine praktikable Lösung, aber die hohe Geschwindigkeit des teuren Parallelmultiplizierers läßt sich gar nicht ausnutzen. Er bildet zwar das Produkt in etwa 100 ns, die wirklich benötigte Zeit wird jedoch dadurch bestimmt, wie lange der Prozessor braucht, um Multiplikand und Multiplikator byte-weise in den Multiplizierer zu laden, das Produkt zu lesen und ins RAM abzulegen.

Zwischen zwei Schreib- oder Leseoperationen vergehen beim 6502-Prozessor 3 µs, in denen die CPU den neuen Operationscode und die neue Adresse liest. Der hier beschriebene Multiplizierer nutzt diese Zeit, um seinen seriell arbeitenden Algorithmus auszuführen. Gegenüber einem Parallelmultiplizierer ist die Geschwindigkeit gleich, der Preis etwa um den Faktor 10 niedriger, und auch der Stromverbrauch ist mit etwa 400 mA wesentlich kleiner. Zur Verwendung kommt der Baustein Am25LS14 von AMD [3, 4]. Dieser Multiplizierer für 8 Bit × 1 Bit verwendet den Booth-'schen Algorithmus [5] zur Multiplikation von Zweierkomplementzahlen. Der Multiplikand wird parallel in 8 Bit Breite geladen. Sodann wird mit jedem Taktimpuls 1 Bit des Multiplikators eingegeben (das LSB zuerst), und am Ausgang steht nach jeder positiven Taktflanke ein neues Bit des Produktes zur Verfügung. Der Am25LS14 läßt sich ohne zusätzliche Logik kaskadieren.

Bild 1 zeigt die Schaltung des Multiplizierers. Drei Vergleicher 74LS85 bestimmen den Adreßbereich, in dem die Schaltung angesprochen werden kann. Die gezeichnete Schalterstellung entspricht dem Adreßbereich 900 X_H. Hierauf beziehen sich die folgenden Adreßangaben.

Der Binärdekoder 74LS138 dekodiert aus den untersten drei Adreßbits 8 Adressen, unter denen auf die verschiedenen Register der Schaltung zugegriffen werden kann. Mit den Adressen 9005 und 9006 kann nur gelesen werden, die übrigen Register lassen sich nur beschreiben.

Den Datenbus puffert der Baustein 74LS245. Aus dem Steuerbus des 6502-Computers wird außer der Schreib/Lese-Umschaltung R/$\overline{W}$ das Taktsignal Φ2 benötigt. Während der Taktphase Φ2 hat sich der Adreßbus stabilisiert, und die Datenübertragung findet statt.

Die Schaltung enthält vier kaskadierte Multiplizierer Am25LS14; sie verarbeitet also 32 Bit breite Multiplikanden. Der Multiplikand wird über die Adressen 9003...9000 byteweise geladen. Das D-Flipflop FF1, das mit dem Datenbit 0 unter der Adresse 9007 gesetzt werden kann, bestimmt über den Mode-Eingang M des ersten Multiplizierers, ob der Multiplikand als Zweierkomplementzahl (Bit 0 = 1) oder als vorzeichenlose Binärzahl (Bit 0 = 0 aufgefaßt werden soll. Der Multiplikator wird unter der Adresse 9004 byteweise in das Schieberegister Am25LS22 geladen.

Das Ende dieses Ladevorgangs triggert über eine RC-Verzögerung einen Impulsgruppen-Generator, der nach [6] aufgebaut ist. Er gibt acht negative Impulse mit einer Folgefrequenz von etwa 6 MHz ab, mit denen die Schieberegister und gegenphasig dazu die Multiplizierer getaktet werden. Das Multiplikatorbyte wird also bitweise an die Y-Eingänge der Multiplizierer gelegt, während gleichzeitig das Produktbyte aus dem S-Ausgang des letzten Multiplizierers in das Schieberegister 74LS299 übernommen wird. Weniger als 2 µs nach dem Laden des Multiplikatorbytes steht das Produktbyte zum Auslesen unter der Adresse 9005 zur Verfügung.

Wenn alle vier Multiplikatorbytes eingegeben worden sind, haben erst die vier niederwertigeren Produktbytes den Multiplikator verlassen. Die vier höherwertigen Ergebnisbytes müssen noch ausgetaktet werden. Dazu dient die Adresse 9006. Sie liest ebenso wie 9005 das Ausgangsschieberegister, nur mit dem Unterschied, daß der Impulsgruppen-Generator getriggert wird. Wenn der Multiplikator als vorzeichenlose Zahl aufzufassen war, muß nun eine „0" an die Y-Eingänge der Multiplizierer gelegt werden. Das macht das Schieberegister Am25LS22 dadurch, daß der serielle Eingang D0 an Masse liegt. Ist der Multiplikator hingegen als Zweierkomplementzahl zu behandeln, so muß nun das Vorzeichen, also das höchstwertige Bit, nachgeschoben werden. Dies erreicht man durch Setzen von FF2 (Bit 1, Adresse 9007), wodurch die „Sign Extend"-Funktion des Am25LS22 aktiviert wird (Bit 7 wird bei jedem Taktimpuls redupliziert). Durch verschiedenes Setzen von FF1 und FF2 ist auch eine gemischte Multiplikation

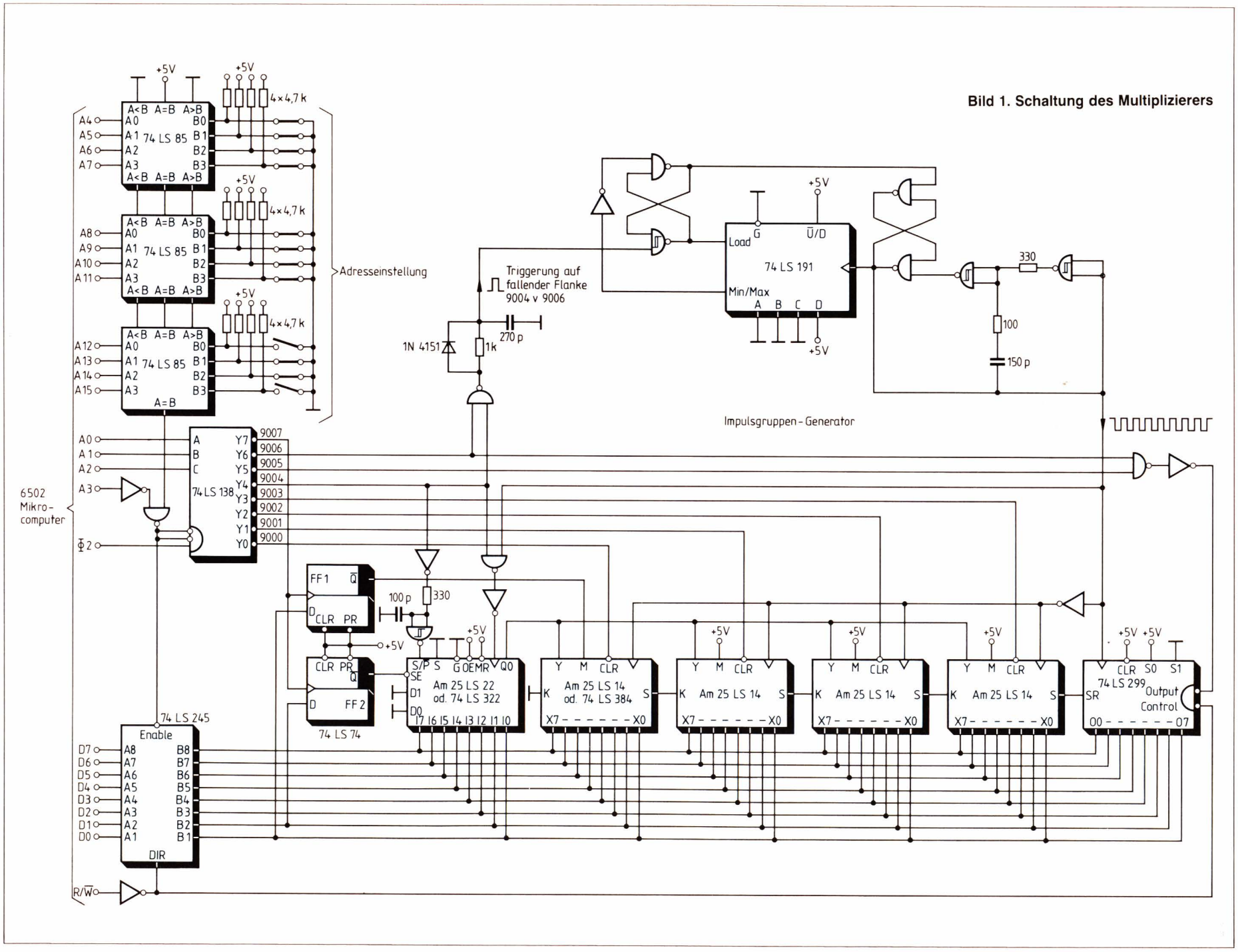


Bild 1. Schaltung des Multiplizierers

möglich; das Produkt hat dann Zweierkomplementdarstellung.

Das Beispielprogramm in *Bild 2* ist linear ohne Verwendung indizierter Adressierungsarten geschrieben, um eine kürzestmögliche Ausführungszeit zu erreichen. Die 32×32-Bit-Multiplikation mit 64 Bit Ergebnis dauert 132 µs. Verzichtet man auf das Lesen und Wegspeichern der niederwertigen Produktbytes, so kann die benötigte Zeit weiter verkürzt werden. Der Multiplikator muß keinesfalls 32 Bit lang sein. Auch eine Operation mit beispielsweise 8 × 32 oder auch 64 × 32 Bit ist möglich, selbstverständlich mit voller Länge des Produktes. Soll der Multiplikand eine von 32 Bit abweichende Breite haben, so sind entsprechend mehr oder weniger Bausteine Am25LS14 zu kaskadieren.

Obwohl der Multiplizierer für einen 6502-Mikrocomputer (AIM 65) entwickelt wurde, dürfte die Anpassung an andere Prozessortypen nicht schwierig sein. Die Frequenz des Impulsgruppen-Generators kann gegebenenfalls durch Verkleinern des 150-pF-Kondensators bis auf etwa 20 MHz erhöht werden, so daß auch beim Anschluß an Prozessoren mit höherer Taktfrequenz keine Wartezyklen nötig werden.

Die Schaltung erschließt den einfachen MOS-Mikroprozessoren auf sehr preiswerte Weise viele Anwendungen der Signalverarbeitung, wie etwa digitale Filterung oder Fouriertransformation. Auch dürfte es sich lohnen, die Gleitkomma-Multiplikationsroutine des Basic-Interpreters umzuschreiben. *Thomas Wittchen*

```
2 zu multiplizierende  Zweierkomplementzahlen
mit 32 Bit Breite muessen sich in den Speicher-
zellen 0200-0203 und 0204-0207 befinden.Das
64 Bit lange Produkt wird in 0208-020F abgelegt.

LDA #03
STA 9007      Bit 0 und 1 gesetzt:
              Multiplikand und Multiplikator
              werden als Zweierkomplementzahlen
              aufgefasst.
LDA 0203      Multiplikand, 1. (LS) Byte
STA 9003
LDA 0202      Multiplikand, 2. Byte
STA 9002
LDA 0201      Multiplikand, 3. Byte
STA 9001
LDA 0200      Multiplikand, 4. (MS) Byte
STA 9000
LDA 0207      Multiplikator, 1. (LS) Byte
STA 9004      Zugriff auf Adresse 9004 triggert
              die Generierung des 1. Produktbytes
LDA 9005
STA 020F      Produkt, 1. (LS) Byte
LDA 0206      Multiplikator, 2. Byte
STA 9004
LDA 9005
STA 020E      Produkt, 2. Byte
LDA 0205      Multiplikator, 3.Byte
STA 9004
LDA 9005
STA 020D      Produkt, 3. Byte
LDA 0204      Multiplikator, 4. (MS) Byte
STA 9004
LDA 9006      Lesen der Produktbytes erfolgt ab
              jetzt von Adresse 9006, um die Gene-
              rierung des naechsten Produktbytes
              auszuloesen, denn von nun an wird
              das Multiplikatorschieberegister
              nicht mehr geladen, sondern schiebt
              durch die "sign extend"-Funktion
              automatisch das Vorzeichen nach.
STA 020C      Produkt, 4. Byte
LDA 9006
STA 020B      Produkt, 5. Byte
LDA 9006
STA 020A      Produkt, 6. Byte
LDA 9006
STA 0209      Produkt, 7. Byte
LDA 9006
STA 0208      Produkt, 8. (MS) Byte
RTS
```

Bild 2. Beispielprogramm

Literatur

[1] *Gerlich, R.:* Zeitbedarf für Rechenoperationen mit dem Mikroprozessor 6502. ELEKTRONIK 1981, H. 7, S. 83.

[2] *Geist, D. J.:* MOS prozessor picks up speed with bipolar multipliers. Electronics, July 7, 1977.

[3] Am25LS14, Datenblatt der Firma AMD.

[4] *Mick, J., Springer J.* und *Ghest, C.:* A high-speed serial/parallel multiplier, the Am25LS14. Druckschrift der Firma AMD.

[5] *Mick, J.:* Mechanization of the Am25LS14 serial/parallel multiplier. Druckschrift der Firma AMD.

[6] *Petersen, K.:* Einfacher Impulsgruppen-Generator. ELEKTRONIK 1977, H. 4, S. 108.

CPU schneller als der Bus

In einem 6502-Europakarten-System benötigte die CPU zu viel Rechenzeit. Eine Erhöhung der Taktfrequenz des gesamten Systems war aufgrund „langsamer" Peripheriekarten und -bausteine nicht möglich. Mit steigender CPU-Taktfrequenz wäre außerdem die Störsicherheit des Gerätes ungünstiger geworden, besonders da sich der Prozessorbus räumlich ausdehnt.

So wurde folgender Weg beschritten: Die CPU wird mit der doppelten Taktfrequenz des Busses betrieben. In einem kleinen räumlichen Bereich arbeiten CPU, RAM und EPROMs mit der vollen Taktfrequenz. Die meisten Zugriffe führt die CPU auf das RAM oder das EPROM durch. Für einen Zugriff auf die langsamen Peripheriekarten wird sie mit dem langsamen Bustakt synchronisiert. Der Bus wird mit der halben Taktfrequenz betrieben. Da die Peripheriekarten selbst auf den Bus zugreifen (wenn $\Phi 2 = 0$ ist, CPU-Zugriff nur bei $\Phi 2 = 1$), ist nur eine konstante Busfrequenz zulässig.

Wird zum Beispiel mit einem LDA-Befehl auf den langsamen Bus zugegriffen (*Bild 1*), ist nur ein Taktzyklus von vieren langsamer. Da die meisten Befehle aber überhaupt nicht auf den langsamen Bus zugreifen, wird der Datendurchsatz nur auf etwa 80 % der vollen Leistung gesenkt (in der Praxis wird dieser Wert oft übertroffen).

Das System wurde mit der CPU 6502B (3 MHz) aufgebaut. Der aus einem Bipolar-PROM bestehende Adreßdecodierer muß das Signal $\overline{LBZ}$ (langsamer Buszugriff) liefern (*Bild 2*). Bei $\overline{LBZ} = 0$ wartet das Flipflop (IC3, Anschl. 2) auf $\Phi 2 = 1$. Dadurch wird der CPU-Takt einen Takt $\Phi 0$ lang auf den langsamen Takt umgeschaltet. Je nach Zeitpunkt des Zugriffs ergibt sich eine der in *Bild 2* dargestellten Möglichkeiten (erster oder zweiter Zugriff). Der langsame Zugriff dauert zwei oder drei Taktzy-

Op-Code/Adr.	Bemerkung
$AD	Op-Code aus EPROM holen
$00	nied. Adreßbyte aus EPROM holen
$20	höh. Adreßbyte aus EPROM holen
$41	Datum aus RAM holen (Buszugriff)

Bild 1. Die vier Zyklen des Befehls LDA $2000. In diesem Beispiel liegt die angesprochene RAM-Adresse (Inhalt $41) auf einer Peripheriekarte. Nur beim letzten Zyklus ist deshalb ein Zugriff über den langsamen Bus erforderlich

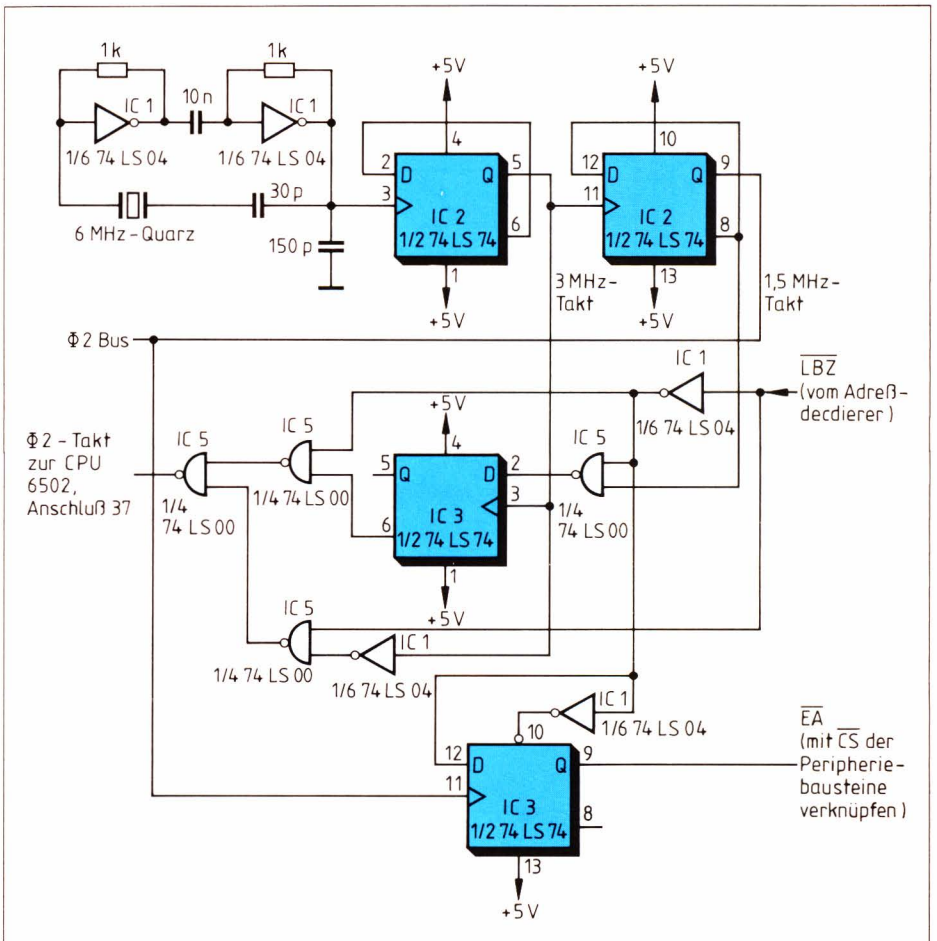


Bild 2. Dieser Taktgenerator sorgt dafür, daß die CPU Buszugriffe mit der halben Taktfrequenz durchführt. Das Signal LBZ muß der Adreßdecodierer liefern, wenn eine Adresse im „langsamen" Bereich liegt

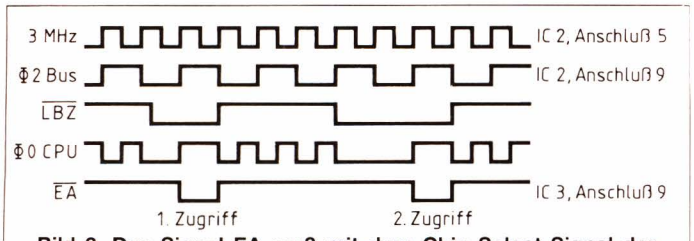


Bild 3. Das Signal EA muß mit dem Chip-Select-Signal der Peripheriebausteine verknüpft werden

klen. Im Fall des zweiten Zugriffs würde ein Adreßdecodierer schon während des ersten Φ2-Bustaktes einen Peripheriebaustein ansprechen. Weil manche ICs solch einen doppelten Zugriff nicht vertragen, wird das Signal $\overline{EA}$ (externe Adresse) gebildet. Es ist nur für die Dauer des eigentlichen Zugriffs auf Null. $\overline{EA}$ muß mit dem Chip-Select der Peripheriebausteine verknüpft werden.

Wegen der gleichen Bussignale läßt sich diese Schaltung auf den Prozessor 6802 übertragen.

Dipl.-Ing. Wolfgang Krebs

Beliebiger Speicherbereich für AIM-65-BASIC

Beim Mikrocomputer AIM-65 ist der Anfang des BASIC-Bereichs auf die Adresse 212 (hex.) festgelegt. Außerdem können nur Programme mit einer Länge von maximal 4 KByte abgespeichert werden. Wer eine Speichererweiterung betreibt, kann mit dem hier vorgestellten Programm den BASIC-Bereich beliebig festlegen.

Der BASIC-Interpreter verwendet bestimmte Zellen, in denen die Grenzen des Speicherbereichs stehen [1]. Diese Zellen sind in der *Tabelle* aufgeführt. Will man den BASIC-Bereich ändern, muß man auch diese „Register" verändern und außerdem an bestimmten Adressen Nullbytes einschreiben. Dies besorgt das Programm BAD (Basic-Adresse), das in *Bild 1* gezeigt ist. Nachdem es abgearbeitet ist, kann man BASIC-Programme in einen beliebigen Speicherbereich laden und ablaufen lassen (in der Praxis von $200...$FFFF).

Die Startadresse des Programms muß auf jeden Fall größer sein als $500, damit es bei der Initialisierung nicht von BASIC-Interpreter überschrieben wird (am besten größer als $1000). Steht es erst einmal im Speicher, dann geht der Benutzer folgendermaßen vor:

Parameter für den BASIC-Bereich des AIM-65

Adresse (hex.)	Inhalt
73...74	Anfangsadresse des Bereiches
75...76	Endadresse des Bereiches
7B...7C	Endadresse von Strings
7F...80	Limit des Bereiches

- BASIC-Interpreter mit Taste „5" wie üblich starten, um die nötigen Daten zu generieren. Die Frage „Memory Size" muß mit 600 beantwortet werden, wenn das Programm BAD im Bereich von $500...$1000 liegt. Sonst ist die Antwort nicht von Bedeutung.
- In den Monitor mit „Escape" zurückspringen.
- Das Programm BAD starten. Es fragt nach dem Programmspeicherbereich (mit FROM = XXXX, TO = XXXX) und springt anschließend in den BASIC-Interpreter zurück (entspricht dem Drücken der Taste „6").

Jetzt kann ein BASIC-Programm in den Rechner eingegeben und gestartet werden. Ein Beispiel zeigt *Bild 2*.

Dipl.-Ing. The Anh Vuong

Literatur

[1] *Butterfield, J.:* Inside Basic. Interactive Nr. 2, Firma Rockwell.

```
; PROGRAMM BAD
; ADDRESS MODIFY
; FOR AIM 65 BASIC
*=$C00
JSR $E7A3 ; FROM=?
LDA $A41C
STA $73
LDA $A41D
STA $74
JSR $E7A7 ; TO = ?
LDA $A41C
STA $7B
STA $7F
LDA $A41D
STA $7C
STA $80
LDY #0 ; NULL BYTE
LDA #0
STA ($73),Y
LDA $73 ; ADD. MODIFY
CLC
ADC #1
STA $73
LDA $74
ADC #0
STA $74
LDA $73
CLC
ADC #2
STA $75
LDA $74
ADC #0
STA $76
LDA #0 ; NULL BYTE
STA ($73),Y
INY
STA ($73),Y
; REENTRY TO BASIC
JMP $B003
BRK
```

Bild 1. Programm zur Änderung des BASIC-Bereichs beim AIM-65

```
<5>
MEMORY SIZE? 600
WIDTH?
 70 BYTES FREE
  AIM 65 BASIC V1.1
  AIM 65 BASIC V1.1

<*>=C00
<G>/
FROM=400 TO=2000

10 I=8^2
20 PRINT I
30 STOP

RUN
 64
 BREAK IN  30
```

Bild 2. Hier wird der BASIC-Bereich auf $400...$2000 festgelegt. Anschließend wird ein Programm eingegeben und gestartet

Watchdog-, Reset- und Zeitbasisschaltung

Störimpulse können jedes Programm „aus der Bahn werfen". Eine Watchdog-Schaltung erkennt den Systemabsturz und sorgt für den geordneten Neustart des Programms. Wir stellen Ihnen eine solche Schaltung vor.

Sie ist als Basis für einfache Steuerungen gedacht. Als Mikrocomputer dient der Typ R6501 (*Bild 1*). Er benötigt nur einen Quarz und ein externes EPROM (bis zu 8 KByte) im Minimalausbau. Die im folgenden beschriebene Hardware kann aber auch an andere Typen angepaßt werden. Die Schaltung benötigt zwei E/A-Leitungen, nämlich PA1 und PA2. Durch Programmieren des Interrupt-Enable-Registers erhält PA1 die Funktion eines positiv flankengetriggerten Interrupt-Einganges (Bit 1 des Interrupt-Enable-Registers auf 1). PA2 ist als Ausgabeleitung vorgesehen.

Zeitbasis

Die Zeitbasis für die Schaltung wird von der Netzfrequenz abgeleitet. Dazu wird die Ausgangsspannung des Transformators mit den Dioden D3 und D4 gleichgerichtet und einem invertierenden Treiber zugeführt. Der Ausgang des Treibers liefert dann bei jedem Nulldurchgang der Trafospannung einen ca. 1 ms langen positiven Impuls mit einer Wiederholfrequenz von 100 Hz. Das Ausgangssignal des Treibers wird der als Interrupt-Eingang programmierten PA1-Leitung zugeführt. D5 verhindert eine Rückwirkung von PA1 auf das Zeitbasissignal.

Resetfunktion

Das Zeitbasissignal löst bei Bedarf auch einen Reset aus. Während der Impulspausen (9 ms) des Zeitbasissignals bekommt der Transistor T1 über D6 und R6 Basisstrom und schaltet damit den Reseteingang des µC auf High-Pegel. Der Zeitbasisimpuls (1 ms) wird über die Diode D6 und das RC-Glied R5/C2 verzögert (40 µs) und öffnet den Transistor T1; der Reseteingang wird dann über R4 auf Low gezogen.

Watchdog

Dieser Resetimpuls wird unterdrückt, wenn es dem µC gelingt, nach dem Interrupt durch PA1, innerhalb der Verzögerung durch R5/C2, eine Überbrückungsspannung an die Basis von T1 zu bringen. Dies geschieht durch ständigen Pegelwechsel von PA2. Der Kondensator C4 lädt sich dabei über C3/D7/D8 auf. Über den invertierenden Treiber fließt dann der Basisstrom von T1 weiter und unterdrückt somit den Resetimpuls. Die Spannung an C4 muß natürlich für die gesamte Dauer des Zeitbasisimpulses aufrechterhalten werden. Dies bedeutet, daß der µC für die Dauer von 1 ms, bei einer Zykluszeit von 10 ms, besetzt ist, also 10 % der Rechenzeit verlorengehen. In vielen Anwendungsfällen ist diese Tatsache jedoch unbedeutend.

Interruptbearbeitung

Der wichtigste Punkt der Interruptbearbeitung ist das Erzeugen des Impulsbursts durch PA2 zum Aufbau der Überbrückungsspannung in der geforderten Zeit. Im Bei-

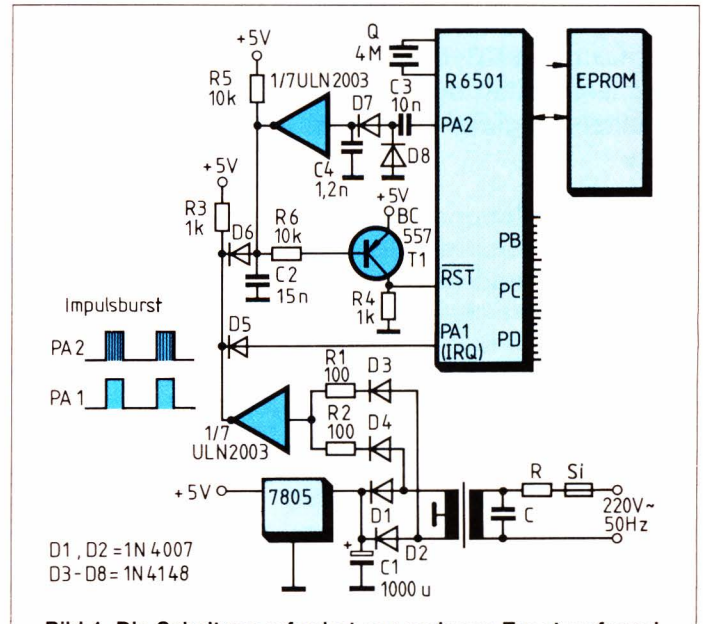


Bild 1. Die Schaltung erfordert nur geringen Zusatzaufwand

```
0006  0000                    EJECT
0007  0000            * INTERRUPT - BEARBEITUNGSPROGRAMM
0008  0000            *
0009  0000            IRQB      START
0010  0000                      USING COMMON
0011  0000            *
0012  0000            DRA       EQU $00      DATA REGISTER A
0013  0000            ICR       EQU $10      INTERRUPT CLEAR REG.
0014  0000            *
0015  0000            * RAM - BELEGUNG
0016  0000            *
0017  0000            FLANK     EQU $40
0018  0000            NTKT      EQU $41
0019  0000            TEST1     EQU $42
0020  0000            TEST2     EQU $43
0021  0000            *
0022  0000                      EJECT
0023  0000 48                   PHA          ACCU RETTEN
0024  0001            *
0025  0001                      SMB2 DRA     BEGINN DES
0026  0003 A93C                 LDA #60      IMPULSBURSTS
0027  0005                      RMB2 DRA
0028  0007 8541                 STA NTKT
0029  0009                      SMB2 DRA
0030  000B A904                 LDA #04
0031  000D                      RMB2 DRA
0032  000F 8540                 STA FLANK
0033  0011            *
0034  0011            LOOP      SMB2 DRA
0035  0013 C641                 DEC NTKT
0036  0015 D003                 BNE CONT1    MAXIMALE DAUER ?
0037  0017 4C1700     TRAP      JMP TRAP
0038  001A            *
0039  001A            CONT1     RMB2 DRA
0040  001C                      SMB1 DRA          ZEITBASIS - SIGNAL
0041  001E                      BBR1 DRA,CONT2    ABFRAGEN
0042  0021 A904                 LDA #04           NACHLAUFZEIT DES
0043  0023 8540                 STA FLANK         IMPULSBURSTS SETZEN
0044  0025 4C1100               JMP LOOP
0045  0028            *
0046  0028 C640       CONT2     DEC FLANK
0047  002A 10E5                 BPL LOOP
0048  002C            *
0049  002C A541                 LDA NTKT     MINIMALDAUER DES ZEITBASIS -
0050  002E 38                   SEC          IMPULSES ÜBERPRÜFEN
0051  002F E932                 SBC #50
0052  0031 9003                 BCC CONT3
0053  0033 4C4200               JMP RETURN   IMPULS WAR ZU KURZ
0054  0036            *                       (STÖRIMPULS)
0055  0036            *
0056  0036 A542       CONT3     LDA TEST1    TESTBYTES ÜBERPRÜFEN
0057  0038 C9AA                 CMP #$AA
0058  003A D0DB                 BNE TRAP
0059  003C A543                 LDA TEST2
0060  003E C955                 CMP #$55
0061  0040 D0D5                 BNE TRAP
0062  0042            *
0063  0042            *         .
0064  0042            *         PROGRAMME FÜR ZEITZÄHLER,
0065  0042            *         AD - WANDLER, USW.
0066  0042            *         .
0067  0042            *
0068  0042 8A         RETURN    TXA          ÜBERPRÜFUNG DER
0069  0043 48                   PHA          RÜCKSPRUNGADRESSE
0070  0044 BA                   TSX
0071  0045 E8                   INX
0072  0046 E8                   INX
0073  0047 E8                   INX
0074  0048 E8                   INX
0075  0049 B500                 LDA $00,X
0076  004B 29F0                 AND #$F0
0077  004D C9F0                 CMP #$F0     HI - ADRESSE DES
0078  004F F003                 BEQ RETNA    ERLAUBTEN ROM BEREICHES
0079  0051 4C1700               JMP TRAP
0080  0054            *
0081  0054            RETNA     RMB1 ICR     LÖSCHE INTERRUPT - FLAG
0082  0056 68                   PLA
0083  0057 AA                   TAX
0084  0058 68                   PLA
0085  0059 40                   RTI
```

Bild 2. Auszug aus dem Interrupt-Bearbeitungsprogramm (Assembler: ORCA/M auf Basis 108)

spiel (*Bild 2*) ist das Retten der Register A und X sowie das Setzen zweier Zähler in die Ausgabe der ersten Impulse eingeflochten. Mit dem Zähler NTKT werden die minimal und maximal erlaubten Längen des Zeitbasisimpulses (an PA1) überprüft. Wird dabei festgestellt, daß die Minimalzeit unterschritten wird, so ist sicher, daß der interruptauslösende Impuls kein Nulldurchgang der Netzwechselspannung ist, sondern ein Störimpuls, und das Interrupt-Bearbeitungsprogramm wird abgebrochen. Beim Überschreiten der Maximalzeit, z. B. bei Netzausfall, springt der Prozessor in eine endlose Schleife (TRAP). Dies hat durch das Ausbleiben des Impulsbursts einen Reset zur Folge.

Nachdem der µC den Impulsburst abgeschlossen hat, ohne daß ein Fehler aufgetreten ist, überprüft er, ob zwei Bytes des RAM-Bereiches (TEST1, TEST2), die am Anfang der Resetsequenz gesetzt wurden, ihre Sollwerte enthalten. Damit kann sichergestellt werden, daß die Resetsequenz ordnungsgemäß durchlaufen wurde oder ob im Fehlerfall der µC bei einer zufälligen Adresse zu arbeiten begonnen hat.

Im Rahmen der Interruptbearbeitung, die ja alle 10 ms erfolgt, ist es möglich, einen oder mehrere Zähler zu inkrementieren und auf diese Art viele voneinander unabhängige Timer für das Hauptprogramm zur Verfügung zu stellen. Auch andere Funktionen wie die Abfrage von A/D-Umsetzern, die Zündung von Triacs oder Thyristoren im Nulldurchgang der Netzspannung usw. können eingebaut werden und sind somit transparent für das Hauptprogramm. Als vorteilhaft hat es sich auch erwiesen, vor jedem RTI-Befehl die Rücksprungadresse, die ja auf dem Stack liegt, zu überprüfen. Damit kann sichergestellt werden, daß der µC vor dem Interrupt im „erlaubten" ROM-Bereich gearbeitet hat.

Aufbau der Hardware

Als invertierender Treiber dient das IC ULN2003. Es werden nur zwei der sieben Invertierer verwendet. Besonders zu beachten ist, daß die Masseleitungen zwischen ULN2003, den Kondensatoren C2 und C4 sowie dem µC so kurz wie möglich sind, um den Einfluß von Netzstörimpulsen zu vermindern.

Die Schaltung ist in Verbindung mit der beschriebenen Software bis jetzt in einigen Dutzend Steuerungen eingesetzt. Über Relais, die der µC über Treiber ansteuert, schaltet sie unter anderem auch induktive Lasten am 220-V-Netz. Außer einem einfachen R/C-Glied am Netzeingang der Steuerung wurden keine weiteren Entstörmaßnahmen getroffen. Trotzdem ist es bis jetzt, nach etwa 1½ Jahren Betriebszeit, noch zu keinem Störfall gekommen. Es kann also gesagt werden, daß dieser Aufbau gegenüber anderen Entstör- und Sicherungsmaßnahmen einen eindeutigen Kostenvorteil bietet und trotzdem einwandfreie Ergebnisse liefert.

Dietmar Geiler

„Watch-dog"-Schaltungen erkennen µP-Systemstörungen

Der Einsatz von Zusatzschaltungen zur Erkennung von Systemstörungen ist immer dann erforderlich, wenn Folgeschäden durch fehlerhaften Betrieb möglich sind. Werkzeugmaschinen-, Aufzugs- oder Ampelsteuerungen sind Beispiele dafür. Dieser Beitrag zeigt Lösungsansätze, die eine einfache Erkennung von Fehlzuständen durch sogenannte Watch-dog-Schaltungen ermöglichen.

1 Störungen in µP-Systemen

Störungen in Mikroprozessorsystemen können durch interne und externe Einflüsse auftreten und eine zeitlich begrenzte oder bleibende Funktionsstörung bewirken. Kritisch sind dabei besonders die Störungen, die fehlerhafte Ausgaben bewirken und somit Folgeschäden auslösen können. Es können folgende Ursachen zu falschen Ausgaben im µP-System führen:

- Fehlerhafte Eingabedaten, verursacht durch externe Störungen oder Bauteileausfälle;
- Bauteileausfälle oder interne Störungen, verursacht durch die endliche Zuverlässigkeit von Bauteilen oder nicht ausgereiften Hardwareentwurf (z. B. Übersprechen usw.);
- Falsche Programmsequenz, verursacht durch Netzstörungen, falsche Eingabedaten, Bauteileausfälle oder versteckte Softwarefehler.

Störungen durch falsche Eingabedaten können mit geeigneten Hardware- und Softwaremaßnahmen, wie z. B. Funktionstestmöglichkeiten der Eingabekanäle (Erkennung von Bauteileausfällen) oder Mehrfachabfragen (Erkennung von externen Störungen), frühzeitig erkannt und unterdrückt werden.

Bauteileausfälle im System sind häufig nur mit hohem Aufwand (Redundanz) erkennbar. Eine günstige Methode, die Funktion der Hardware zyklisch zu überprüfen, ist der Einsatz von Selbstdiagnostikprogrammen [1]. Besondere Beachtung ist der Störsicherheit des Netzteils zu schenken, denn hier verursachen Störungen oft eine falsche Programmsequenz („der Mikroprozessor läuft in den Wald") und somit (meist nicht erklärbare) Systemstörungen. So ist der Einsatz von guten Netzfiltern, abgeschirmten Transformatoren, getakteten Netzteilen mit Spannungsreserve, Netzausfallerkennungs- und Spannungsüberwachungsschaltungen besonders zu empfehlen. Auch sollte ein Störungstest mit geeigneten Netzstörsimulatoren niemals ausgelassen werden, um spätere unliebsame Überraschungen zu vermeiden.

Um die Kosten in Hard- und Software in einem wirtschaftlich vertretbaren Rahmen zu halten, ist der Einsatz von Zusatzschaltungen, z. B. Watch-dog-Schaltung, empfehlenswert.

2 Was ist eine Watch-dog-Schaltung?

Die wörtliche Übersetzung der englischen Bezeichnung Watch-dog ist Wachhund und deutet bereits an, daß die Schaltung etwas be- oder überwachen soll. Bei der Anwendung in Mikrocomputersystemen versteht man darunter in der Regel, daß die Schaltung den Programmablauf oder die Ausgabedaten überwacht. Bei Erkennung von Fehlerzuständen werden dann Aktionen eingeleitet, welche einen definierten Anfangszustand herstellen, um das System erneut zu starten, zu blockieren oder falsche Ausgaben zurücksetzen.

Der Aufwand in Hard- und Software ist abhängig von der gewünschten Reaktionszeit und dem Sicherheitsgrad, mit dem Fehlzustände erkannt werden.

Am Beispiel einer Ampelanlage läßt sich dies sehr gut erklären. Kommt es durch Störungen zu fehlerhaften Zuständen der Ampel, so wirken sich diese durch die Reaktionszeit der Verkehrsteilnehmer erst frühestens nach 200 ms aus. Dies bedeutet für die Watch-dog-Schaltung, daß eine kombinierte Hardware- (z. B. ein retriggerbares Monoflop SN74LS123N) und Softwarelösung mit geringem Aufwand möglich ist.

Nachfolgend werden die Realisierung des sogenannten Watch-dog-Zeitgebers mit Reaktionszeiten im Millisekundenbereich und die Watch-dog-Sperre mit Zeiten im µs-Bereich, beschrieben.

3 Der Watch-dog-Zeitgeber

Die einfachste Möglichkeit einer Watch-dog-Schaltung ist der Zeitgeber, der mit dem System-Reset gestartet wird und vom Programm zyklisch vor Ablauf der eingestellten Zeit zurückgesetzt werden muß. Mit dem Rücksetzen des Zeitintervalls wird dieses automatisch wieder gestartet.

3.1 Schaltungsaufbau

Die Schaltung im *Bild 1* zeigt den Aufbau eines zweistufigen Watch-dog-Zeitgebers, der mit dem re-

triggerbaren Monoflop SN74LS123N aufgebaut ist. Zweistufig deshalb, weil nach nichterfolgtem Rücksetzen des ersten Zeitintervalls t_1 ein erneuter Systemstart über einen nichtmaskierbaren Interrupteingang, wie z. B. den LOAD-Eingang, versucht wird. Mit dem Ablauf der ersten Monoflop-Zeit (MF1 in Bild 1) wird ein zweites Monoflop (MF2) gestartet. Bleibt der Restart-Versuch erfolglos, wird nach Ablauf von t_2 ein HOLD ausgelöst, der das System blockiert. Um eine ausreichende Systemflexibilität zu erlangen, sind in der Schaltung Steckbrücken vorgesehen. Mit Ablauf der ersten Zeit t_1, wird automatisch mit OUTCLR der Ausgabeport zurückgesetzt. *Bild 2* zeigt den Zeitablauf bei korrekter und fehlerhafter Arbeitsweise des Mikrocomputersystems.

3.2 Programmierung

Bei der Programmerstellung muß sichergestellt werden, daß vor Ablauf der Zeit t_1 (Bild 2) das Monoflop wieder neu gestartet wird. Kommt der Mikroprozessor dann durch Systemstörungen „aus dem Tritt", wird das System neu gestartet. Diese einfache Methode schützt vor einer „Schleife ohne Ende", die z. B. durch Ausführung nicht vorhandener Speicheradressen vorkommt. Kehrt der Mikroprozessor nach Auftreten der Störung in das normale Programm zurück, wird der Zeitgeber eventuell wieder zurückgesetzt. Besitzt der Mikroprozessor einen Befehlsvorrat der positionsunabhängige Programmausführung erlaubt (*position independent program execution*), kann diese weitgehend softwaremäßig verhindert werden. *Bild 3* zeigt die grobe Programmstruktur für den Typ TMS9900. Das Rücksetzen des Watch-dog-Zeitgebers ist hier als Unterprogramm (Aufruf mit BL@TIMOUT) aufgebaut. Das Unterprogramm prüft vor dem Rücksetzen des Zeitgebers, ob in einem vereinbarten Speicherplatz (LASTPC) der gleiche Inhalt steht, wie in der Adresse, die dem Unterprogrammaufruf folgt (LPC(n-x)). Bei der Initialisierung des Anwenderprogrammes lautet die Befehlssequenz für den TMS 9900:

	LI	RX, OPRT	Lade B * R11 Rücksprung
	BL	RX	Führe Rücksprung in RX aus
LPCn	MOV	R11, @LASTPC	Speichere LPCn

Die absolute Adresse von LPCn wird in der Adresse LASTPC abgelegt. Nach dem Laden des Rücksprungs (B * R11) in eines der 16 TMS9900-Register (hier bezeichnet als RX) wird ein Unterprogrammsprung (BL RX) nach RX ausgeführt. Die Ausführung dieses Befehls bewirkt die Abspeicherung der Rücksprungadresse (LPCn) nach R11, und die Ausführung des B * R11 in RX bewirkt den Rücksprung zur Marke LPCn. Die absolute Adresse von LPCn steht jetzt in R11 und wird mit „MOV R11, @LASTPC" unter der Adresse LASTPC abgelegt [2].

Vor Ablauf der Zeit t_1 wird dann das Unterprogramm aufgerufen. Es vergleicht, ob LASTPC den gleichen Inhalt aufweist wie das Wort, das dem „BRANCH-AND-LINK"-Aufruf folgt. Ist der Vergleich negativ, so liegt eine falsche Programmsequenz vor.

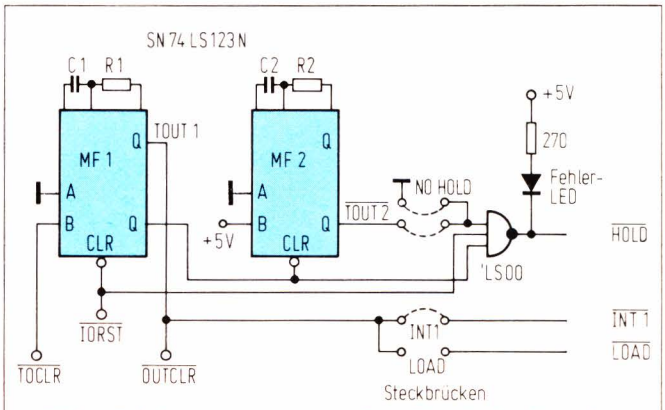


Bild 1. Schaltung eines zweistufigen Watch-dog-Zeitgebers

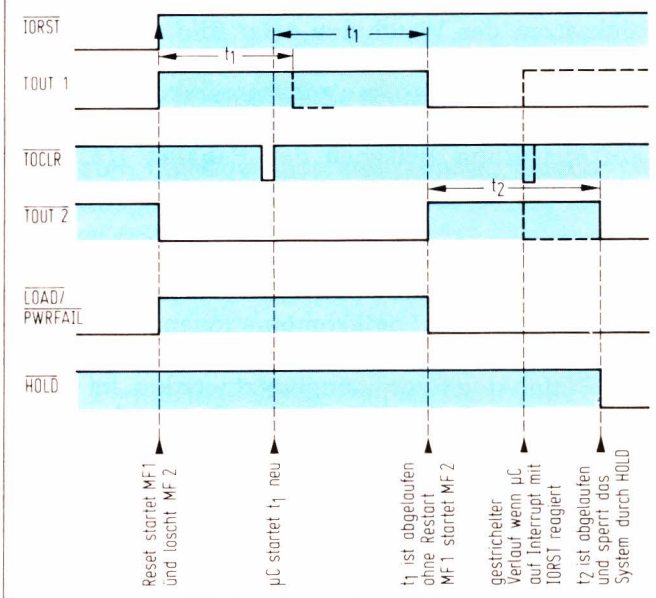


Bild 2. Zeitdiagramm des Watch-dog-Zeitgebers

```
        BL    TIMOUT          RUFE TIMEOUT ROUTINE AUF(Zeitpunkt tn-1)
        DATA  LPC (n-1)       LETZTE GENERIERTER PC-ADRESSE
                                                                  ┐
          ┬                   }                                   │
          ANWENDER            }ca. t 1/2 Zeitabstand              │
          PROGRAMM            }                                   │
          ┴                   }                                   │
        LI    RX,OPRT         LADE  B  R11 Rücksprung             │
        BL    RX              FÜHRE RÜCKSPRUNG IN RX AUS          │
LPCn    MOV   R11  LASTPC     SPEICHERE LPCn                      │
          ┬                   }                                   │
          ANWENDER            }ca. t 1/2 Zeitabstand              │ t1min
          PROGRAMM            }                                   │
          ┴                   }                                   ┘
        BL    TIMOUT          RUFE TIMOUT-RÜCKSETZ ROUTINE AUF
        DATA LPCn
          ┬
          ANWENDER
          PROGRAMM
          ┴
* ROUTINE ZUR ÜBERPRÜFUNG DER LETZTEN ÜBERGEBEN PC-ADRESS UND
* RÜCKSETZEN DES "WATCH-DOG"-TIMERS
TIMOUT  C  R11+,  LASTPC    LETZTER GESPEICHERTER PC-WERT OK?
        JNE  ERROR          SPINGE ZUR FEHLERBEHANDLUNG
        SBZ  TOCLR          SETZE TIMEOUT ZURÜCK
        RT                  SPRINGE ZUM NORMALPROGRAMM ZURÜCK
ERROR   :                   FEHLERBEHANDLUNG
        :
```

Bild 3. Programmsequenz zur Überprüfung des Ablaufs und Rücksetzen des „Watch-dog-Zeitgebers"

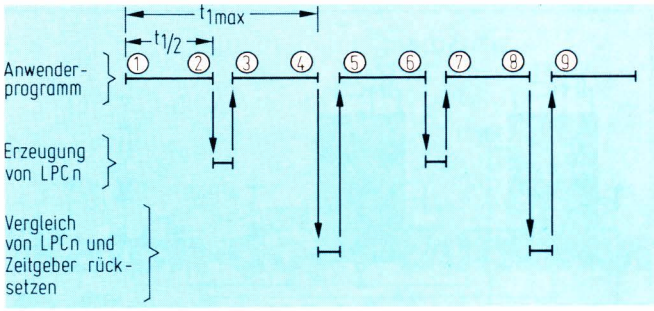


Bild 4. Zeitablauf der sequentiellen Adreßerzeugung, des Vergleichs und des Rücksetzens des Watch-dog-Zeitgebers

Den zeitlichen Ablauf des Anwenderprogramms, der Erzeugung der sequentiellen Adresse (LPCn) und dem Unterprogramm zum Vergleich der Adressen und Rücksetzen des Watch-dog zeigt *Bild 4*. Die Wahrscheinlichkeit, daß eine Systemstörung auftritt, die ein Verlassen des Anwenderprogramms *und* die Rückkehr in den gleichen Abschnitt bewirkt (Ein/Aussprung zwischen den Punkten 1/2,3/4,5/6 usw. nach Bild 4) ist als gering einzustufen.

4 Die Watch-dog-Sperre

Wird eine schnelle Reaktionszeit benötigt und sollen unerlaubte Logikkombinationen (verursacht durch Systemstörungen) vermieden werden, kann eine Watch-dog-Sperre eingesetzt werden. Im Gegensatz zum Watch-dog-Zeitgeber muß sie nicht zurückgesetzt werden.

4.1 Schaltungsaufbau

Bild 5 zeigt eine solche Schaltung für ein 8-Bit-Ausgabeport des TMS9900, mit einer Reaktionszeit von 1 µs.

Die Schaltung arbeitet mit einem PROM SN74S387N (256 x 4 Bit) im Auswertezweig, mit der alle unerwünschten Ausgangskombinationen des Port erkannt werden. Die vier PROM-Ausgänge liefern die Signale zur Rücksetzung des Ports oder erzeugen die Interrupt-, LOAD- oder HOLD-Signale.

Das 4-Bit-Register SN74LS174N dient zur Synchronisation und Unterdrückung von störenden Signaländerungen des PROM während der Umschaltung der Eingänge.

Die Watch-dog-Sperre erfordert einen geringeren Softwareaufwand als der Watch-dog-Zeitgeber. Ihr Nachteil ist, daß der Faktor Zeit nicht berücksichtigt wird. Systemstörungen werden nur dann erkannt, wenn sie zu unerlaubten Ausgangskombinationen führen. Ein Einsatz bei Ausgabeports mit paralleler 8-Bit-Ausgabe ist nicht möglich.

5 Hinweise für den Aufbau von µP-Ausgabemoduln

Der Entwurf von Ausgabemoduln für Mikroprozessorsysteme sollte die Verwendung von überwachenden Zusatzschaltungen beinhalten. Eine entsprechende Steckbrückenprogrammierung erlaubt dann den flexiblen Einsatz während der Systemrealisierung. Zur Überprüfung der Funktionstüchtigkeit der Ausgabekanäle empfehlen sich außerdem parallelgeschaltete Eingabeports. Dabei sollte das Eingabeport möglichst nahe am Stellglied angeschlossen sein. Dies gestattet dann die Überprüfung der Ausgabedaten durch erneutes Einlesen (*echo back*) und Operandenvergleich. Es ist außerdem von Vorteil, optische Anzeige für erkannte Fehlfunktionen vorzusehen. Sie erhöht die Wartbarkeit und vereinfacht den Service.

6 Schlußbemerkungen

Die diskutierten Schaltungen stellen nur eine Auswahl der vielen Möglichkeiten dar. Entsprechend den Anforderungen sind auch programmierbare Zeitgeber (z. B. TMS9901) oder programmierbare Teiler einsetzbar. Wird eine hohe Sicherheit gefordert, kann ein zweiter Mikroprozessor zur Überprüfung der Ausgabedaten herangezogen werden.

Ing. (grad.) Horst Huse

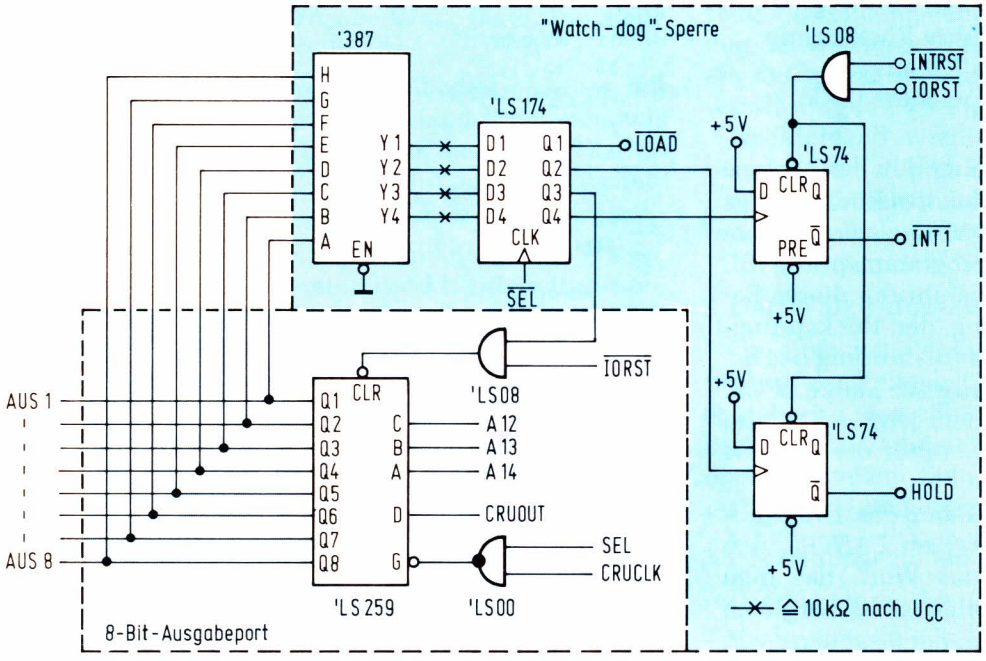


Bild 5. Schaltungsbeispiel der Watch-dog-Sperre

Literatur

[1] EPROM-Prüfsummenprogramm für das TM990-Mikrosystem. Applikationsbericht (EB 129) der Firma Texas Instruments, Deutschl. GmbH.

[2] Adreßunabhängige EPROM-Software. Applikationsnotiz (EN 6) der Firma Texas Instruments, Deutschland GmbH.

[3] Funktionen und Anwendungen des programmierbaren Systeminterface-Bausteins TMS 9901. Applikationsbericht (EB 123) der Firma Texas Instruments, Deutschl. GmbH.

Mikrocomputer fragt Multifunktionszähler ab

Der bekannte Baustein ICM7226 besticht nicht nur durch die Vielzahl der Funktionen, die er ausführen kann, sondern auch durch eine Reihe von Ausgängen, die einen Rechneranschluß ermöglichen. Im vorliegenden Fall sollte lediglich zu einem vom Rechner bestimmten Zeitpunkt die Frequenzinformation ausgelesen werden. Wesentlich komplexere Aufgaben, bis hin zum IEC-Bus-Anschluß, können jedoch mit Hilfe der Ausgänge $\overline{\text{MEAS}}$, $\overline{\text{STORE}}$ und RESET auf ähnliche Weise gelöst werden.

Das Herz des Chips [1] ist ein 8-Stellen-Zähler, der seine Information im BCD-Code in ein Latch schreibt, aus dem sie seriell ausgelesen und in das 7-Segment-Format gewandelt wird (beide Formate stehen an Ausgängen zur Verfügung, *Bild 1*). Ausgelesen wird mit einer Rate von 500 Hz (abgeleitet aus der Quarz-Basis von 10 MHz); gleichzeitig wird über einen 3-zu-8-Decodierer/-Treiber ein Digit-Impuls erzeugt. Die Logik, die über diese Digit-Impulse gesteuert wird, ermöglicht nun eine ganze Reihe von Funktionen und Bereichen. Im einzelnen sind das am FUNCTION-Eingang: Frequenz- und Periodenmessung, Frequenzverhältnis Eingang A zu B, Zeitintervall (zwischen Impuls an A und Impuls an B), Ereigniszählung und Darstellung der Oszillatorfrequenz. Am RANGE-Eingang kann die Zeitbasis von 0,01 s bis 10 s eingestellt werden (im 1-s-Bereich beträgt die Auflösung 1 Hz bei 10 MHz Endwert), die Zeit zwischen zwei Messungen beträgt 200 ms. Der zusätzliche CONTROL-Eingang ermöglicht eine Reihe von Tests sowie die Auswahl einer 1-MHz-Basis bzw. einer belie-

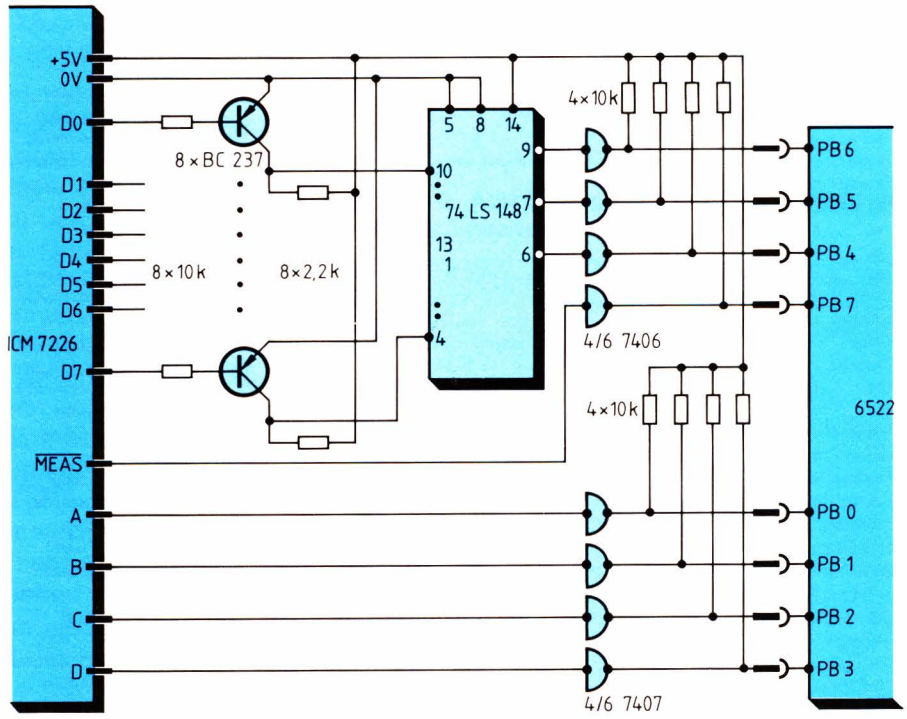


Bild 1. Anpassungsschaltung zwischen Zählerbaustein und Mikrocomputer. Hinweis: Die Schwelle am Eingang des Zählers liegt bei 2 V (bei 5 V Versorgungsspannung). Das Tastverhältnis sollte 50 % betragen. Zur Umformung anderer Signalformen als Rechteck sollte ein schneller Komparator wie z. B. NE 526 eingesetzt werden. Betreibt man an den Segment-/Digit-Ausgängen 8-mm-LEDs (MAN 71A), so liegt die Stromaufnahme bei 0,5 A

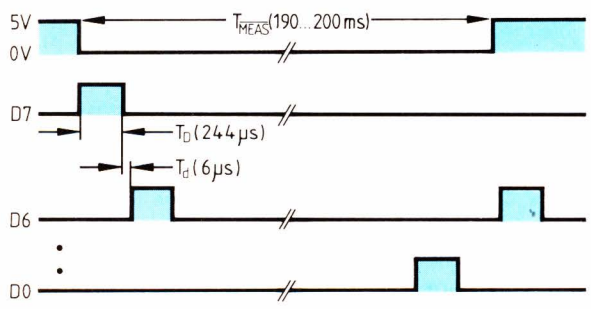


Bild 2. Zeitbeziehung zwischen den Digit-Informationen

bigen externen Basis. Ebenso kann der Dezimalpunkt extern gesetzt werden. Der Ablauf der Messung kann über HOLD und $\overline{\text{RESET}}$ gesteuert werden. Die Zeitbasis läßt sich über den EXT.-RANGE-Eingang beliebig einstellen. Der Chip stellt durch die Ausgänge $\overline{\text{MEAS}}$, $\overline{\text{STORE}}$ und RESET weitere Information zum Ablauf zur Verfügung. Es sind also auch komplexe Rechnersteuerungen wie z. B. IEC-Bus-Anschluß möglich.

Der Zähler stellt die Ziffern-Information direkt im BCD-Code dar. Es muß also nur aus der Digit-Information der LED-Anzeigeeinheit eine entsprechende rechnerkompatible Information abgeleitet werden (*Bild 2*). Dies geschieht durch Pegelanpassung und 8-zu-3-Codierung im Baustein 74LS148. Zusammen mit der Gültigkeitsinformation ($\overline{\text{MEAS}}$, *measurement in progress*) erhält man also ein 8-Bit-Wort, das von jedem gängigen Mikroprozessor über einen Parallel-Port übernommen werden kann. In der realisierten Schaltung wurde noch ein Puffer für die BCD-Ziffer eingefügt.

Das Programm zum Auslesen der Frequenz-Information muß nun die Gültigkeit ($\overline{\text{MEAS}}$) prüfen und dann 8mal die BCD-Information mit der jeweiligen Digit-Information übernehmen und abspeichern. Das Listing zeigt die Realisierung mit dem Mikroprozessor 6502 (AIM-65).

Nach der $\overline{\text{MEAS}}$-Prüfung wird aus der Digit-Information ein Offset erzeugt und damit die BCD-Information abgespeichert, das nächste Digit abgewartet und die Übernahme wiederholt. Das Unterprogramm DISP gibt die Information fortlaufend auf dem Display und dem Drucker – wenn dieser aktiviert ist – aus. Wird in Zeile 0223 stattdessen in das Unterprogramm PACK gesprungen, so wird die Information als 2-Byte-Wort an einer Speicherstelle niedergelegt, auf die dann andere Programme zugreifen können. Ein Aussprung zum Monitor über die Taste ESC bzw. das Anhalten des Programmes über SPACE ist ebenfalls möglich (RCHECK).

Thomas Lemmer

```
010F          4C0002 JMP INIT
0111          ;
0111  CLR            =$EB44
0111  OUTDP          =$EEFC
0111  RCHECK         =$E907
0111  ORB            =$A000
0111  DDRB           =$A002
0111                 +=$0200
0200
0200          ;+++++ Hauptprogramm ++++++++++++++++++++++++++++++++++++
0200  INIT    A000   LDY #00          ;Initialisiere Y-Reg
0202          78     SEI
0203          A900   LDA #$00         ;Init. DDRB
0205          8D02A0 STA DDRB
0208  START   AD00A0 LDA ORB          ;Warte auf MEAS
020B          30FB   BMI START
020D          29F0   AND #$F0         ;Maskiere Digits und erzeuge daraus
020F          8D8902 STA STODIG
0212          4A     LSR A            ;Die Digit-Nummer
0213          4A     LSR A
0214          4A     LSR A
0215          4A     LSR A
0216          AA     TAX
0217          AD00A0 LDA ORB
021A          29 0F  AND #$0F         ;Maskiere BCD-Ziffer
021C          9D8102 STA STOBCD ,X    ;Und lege sie an der Digit-
021F          C8     INY              ;Adresse ab
0220          98     TYA
0221          C908   CMP #$08         ;Das Ganze 8mal
0223          F02A   BEQ DISP         ;Aussprung in die Anzeige-Routine
0225  CHANGE  AD00A0 LDA ORB
0228          29F0   AND #$F0         ;Im folgenden wird auf Wechsel des
022A          CD8902 CMP STODIG       ;Digits geprüft, dann MEAS abge-
022D          F0F6   BEQ CHANGE       ;Wartet und die nächste Ziffer
022F          4C0802 JMP START        ;Übernommen
0232          ;+++++ Subroutinen ++++++++++++++++++++++++++++++++++++
0232  PACK    58     CLI
0233          A000   LDY #$00         ;Packt zwei BCD-Ziffern in ein
0235          A207   LDX #$07         ;Byte
0237  SHIFT   BD8102 LDA STOBCD,X
023A          0A     ASL A
023B          0A     ASL A
023C          0A     ASL A
023D          0A     ASL A
023E          CA     DEX
023F          1D8102 ORA STOBCD,X
0242          CA     DEX
0243          998A02 STA STOPA,Y
0246          C8     INY
0247          98     TYA
0248          C904   CMP #$04
024A          D0EB   BNE SHIFT
024C          4C7902 JMP LOOP1
024F  DISP    58     CLI              ;Ausgabe auf Display/Drucker
0250          A207   LDX #$07
0252          A000   LDY #$00
0254  TRANSF  A930   LDA #$30         ;Transformiert BCD in ASCII
0256          1D8102 ORA STOBCD,X
0259          999702 STA DISPLA,Y
025C          C8     INY
025D          CA     DEX
025E          98     TYA
025F          C908   CMP #$08
0261          D0F1   BNE TRANSF
0263          2044EB JSR CLR
0266          A2FF   LDX #$FF
0268 LOOP     E8     INX              ;Ausgabeschleife
0269          BD8F02 LDA DISBUF,X
026C          C93B   CMP #';'
026E          F006   BEQ CHECK
0270          20FCFF JSR OUTDP
0273          4C6802 JMP LOOP
0276 CHECK    2007E9 JSR RCHECK       ;Hier kann zum Monitor zurück
0279 LOOP1    AD00A0 LDA ORB          ;Gesprungen werden
027C          10FB   BPL LOOP1        ;Hier wird wieder MEAS geprüft
027E          4C0802 JMP START        ;Und zurück an den Anfang gesprungen
0281 STOBCD   3030   .BYT '00000000'
0289 STODIG   30     .BYT '0'
028A STOPA    3030   .BYT '0000'
028E DISBUF   4652   .BYT 'FREQUENZ 00000000 HZ;'
02A3                 +=+-12
0297 DISPLA   3030   .BYT '00000000'
```

Adressenextensor für 8-Bit-µP

Gebräuchliche Mikroprozessoren mit einer Wortbreite von 8 Bit haben in der Regel einen Adreßbus von maximal 16 Bit, mit dem man dann 65536 Speicherzellen direkt adressieren kann. Bei der Verarbeitung großer Datenmengen oder im Zusammenhang mit höheren Programmiersprachen (z. B. PASCAL) weicht man zur Ausdehnung des Speicherraumes meist auf periphere Hintergrundspeicher (z. B. Floppy Disk) aus.

Speichererweiterung mit RAM

Eine andere, bereits praktizierte Möglichkeit, ist die, den zu eng gewordenen Adreßbereich des µP durch geeignete Hardware- und Softwaremaßnahmen zu erweitern. Dazu werden weitere Speicherkarten prinzipiell am µC-Bus belassen, müssen aber vom µP-Adreßbus getrennt und mit softwaremäßig gesteuerten externen Adressen und Selektionssignalen versorgt werden. Auf der Hardwareseite ist dies mit Portbausteinen realisierbar, bedingt aber einen hohen Bauteile- und Verdrahtungsaufwand.

Lösung mit „Extensor"

Nachdem ein Abbild dieser externen Adressen zu gewissen Zeitpunkten zuvor auch auf dem µP-Adreßbus erscheint, bietet es sich an, durch einen geeigneten „Extensor" (= „Streckmuskel") diese Adreßinformation auch auszunutzen, d. h. zusätzliche Speicherkarten mit ihren Adressen auch am Bus zu betreiben und nur zu diesen Zeitpunkten anzuwählen. Gleichzeitig ist natürlich der Systemspeicher zu desaktivieren.

Da ein Deselektionssignal meist schon auf dem Systembus definiert ist, sind lediglich noch einzelne Leitungen (oder auch verdrillte Zweidrahtleitungen mit Masse) auf die Erweiterungskarten zu führen, falls diese Leitungen auf dem Bus nicht Platz finden.

Analysiert man die Information auf dem Adreßbus zu verschiedenen Zykluszeiten des µP, so fällt auf, daß bei Lese- (z. B. LDA) und Schreibanweisungen (z. B. STA) die Speicheradresse während einer bestimmten Zykluszeit auf dem Adreßbus liegt. *Tabelle 1* zeigt, daß dies für einen angenommenen µP (Bus nicht im Multiplex geschaltet) während Zyklus i+3 der Fall ist.

Will man erreichen, daß diese Adresse nicht für das System-RAM gilt, so muß es in diesem und nur diesem Zyklus deselektiert werden. Das zu diesem Zweck zu erzeugende Signal kann gleichzeitig zur Selektion eines weiteren 64-K-Bereiches herangezogen werden. Sollen dagegen mehrere 64-K-Blöcke angesprochen werden können, so wird dieses Signal mit weiteren Adreßinformationen verknüpft.

Tabelle 1. Adreßbusinhalt eines hypothetischen µP für einen Speicherbefehl

OPCODE	Adr.-Bus	Zyklus	Bemerkung
LDA/STA	PC	i	PC = Programmzähler
	PC+1	i+1	
	PC+2	i+2	
	ADH,L	i+3	Speicheradresse

Tabelle 2. Adreßbusinhalt für Pseudobefehl und Speicherbefehl in erweitertes RAM

Schritt	OPCODE	Adr.-Bus	Zeit	Funktion, Bemerkung
1	STA	PC	t	Pseudobefehl
		PC+1	t+1	
		PC+2	t+2	
		ADH,L + (INDEX)	t+3	Initialisierungsadresse, Index wählt 64-K-Bank, Zähler und Latch gesetzt, Zählerstand = 11
2	LDA/STA	PC+3	t+4	Zählerstand = 12
		PC+4	t+5	Zählerstand = 13
		PC+5	t+6	Zählerstand = 14
		ADH,L + (INDEX)	t+7	Speicheradresse für erweitertes RAM, indiziert, 64-K-Bank selektiert, System-RAM deselektiert
3	STA/XXX	PC+6	t+8	Zählerstand = 15 Zähler hält an

Die Initialisierung dieses Selektionsvorganges muß natürlich vor der eigentlichen Schreib-/Leseoperation im erweiterten Speicher erfolgen. In der Praxis geschieht die Initialisierung nun durch einen vorgeschalteten Pseudobefehl mit indizierter Adresse als Operanden. Die Adresse darf in einem ROM-Bereich liegen, um keinen zusätzlichen Adreßraum zu verbrauchen. Dabei muß man allerdings vermeiden, daß bei einer Leseoperation aus dem ROM die Fremdselektion initialisiert wird. Durch die Wahl eines Schreibbefehls (z. B. STA) als Pseudobefehl und die Verknüpfung des Deselektionssignals mit dem Schreibtakt des µP ist dies leicht möglich. Die dann vorgeschriebene Schreiboperation hat zudem den Vorteil, daß der Akkumulatorinhalt bei der Ausführung des Pseudobefehls nicht verloren geht.

Tabelle 2 zeigt den zeitlichen Ablauf für den hypothetischen µP. In Schritt 1 wird durch den Pseudobefehl zum Zeitpunkt t+3 der Extensor initialisiert und eine 64-K-Bank über den Inhalt des Indexregisters angewählt. Der in Schritt 2 darauf folgende Schreib- bzw. Lesebefehl für das erweiterte RAM zum Zeitpunkt t+4 muß in beiden Fällen die gleiche Anzahl von Zyklen aufweisen, da diese in die Hardware-Steuereinheit eingeht, deren Prinzipschaltbild im *Bild* gezeigt ist.

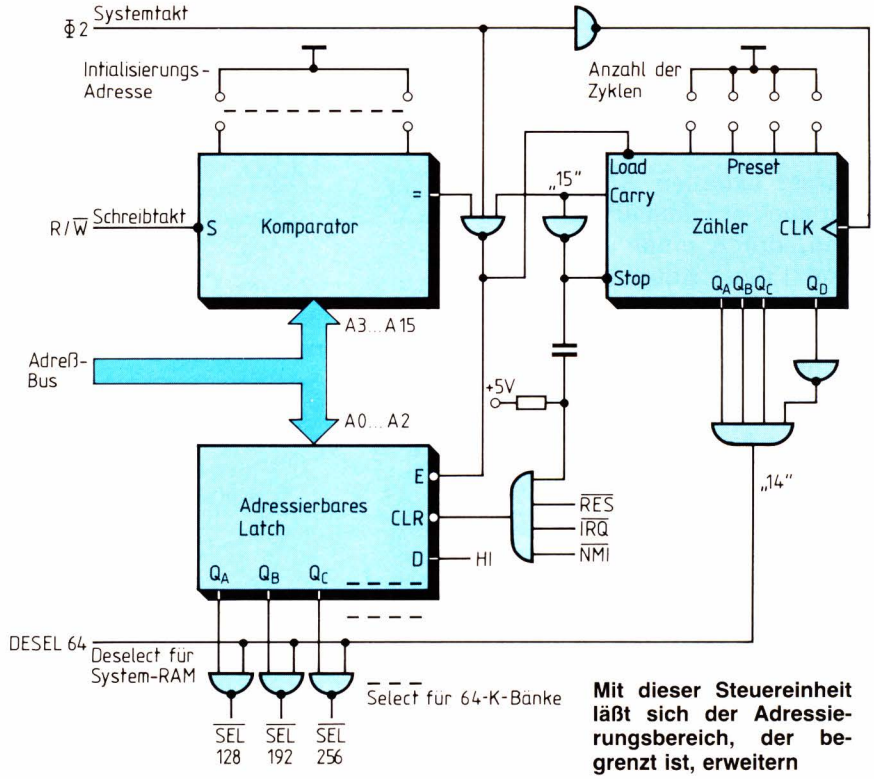
Für das Beispiel in Tabelle 2 ist der Zähler zur Zeit t+3 mit 11 vorzusetzen. Er zählt dann mit jedem Zyklus um 1 aufwärts bis 14. Zu genau diesem Zeitpunkt t+7 liegt nun auf dem Adreßbus des µP die Adresse für die in t+3 gewählte 64-K-Bank, die gleichzeitig selektiert wird, während der µP von seinem System-RAM getrennt wird. Im nächsten Zyklus t+8 wird der Zähler mit dem Inhalt 15 angehalten, die Bank deselektiert und der Prozessor kann wieder über seinen Speicher verfügen, um in Schritt 3 im Programm fortzufahren.

Auf ein Detailschaltbild soll hier verzichtet werden, da es zu systemspezifisch wäre. In einem Prototypen für ein 6502-System wurden folgende Bausteine verwendet:
- 74LS161 als Zähler;
- 74LS259 für das adressierbare Latch;
- Am 25LS2521 als Adreßkomparator;
- 74LS-Gatterbausteine.

Anwendung

Ein Ausbau der Steuereinheit ist einfach. Für jeden zusätzlichen 64-K-Speicherbereich werden nur ein Latch und eine Adresse im System-Adreßraum benötigt.

Angesichts steigender Integrationsdichte und fallender Preise bei Speicherbausteinen ist nicht einzusehen, warum man sich mit einem 64-K-Speicherbereich zufrieden geben soll, nur weil man einen 8-Bit-µP betreibt. So laufen z. B. unter einem µP 6502 bereits 256 KByte RAM auf einer Einfacheuropakarte, in dem zwei Floppy-Disk-Laufwerke emuliert werden. *Rolf Schöne*



Mit dieser Steuereinheit läßt sich der Adressierungsbereich, der begrenzt ist, erweitern

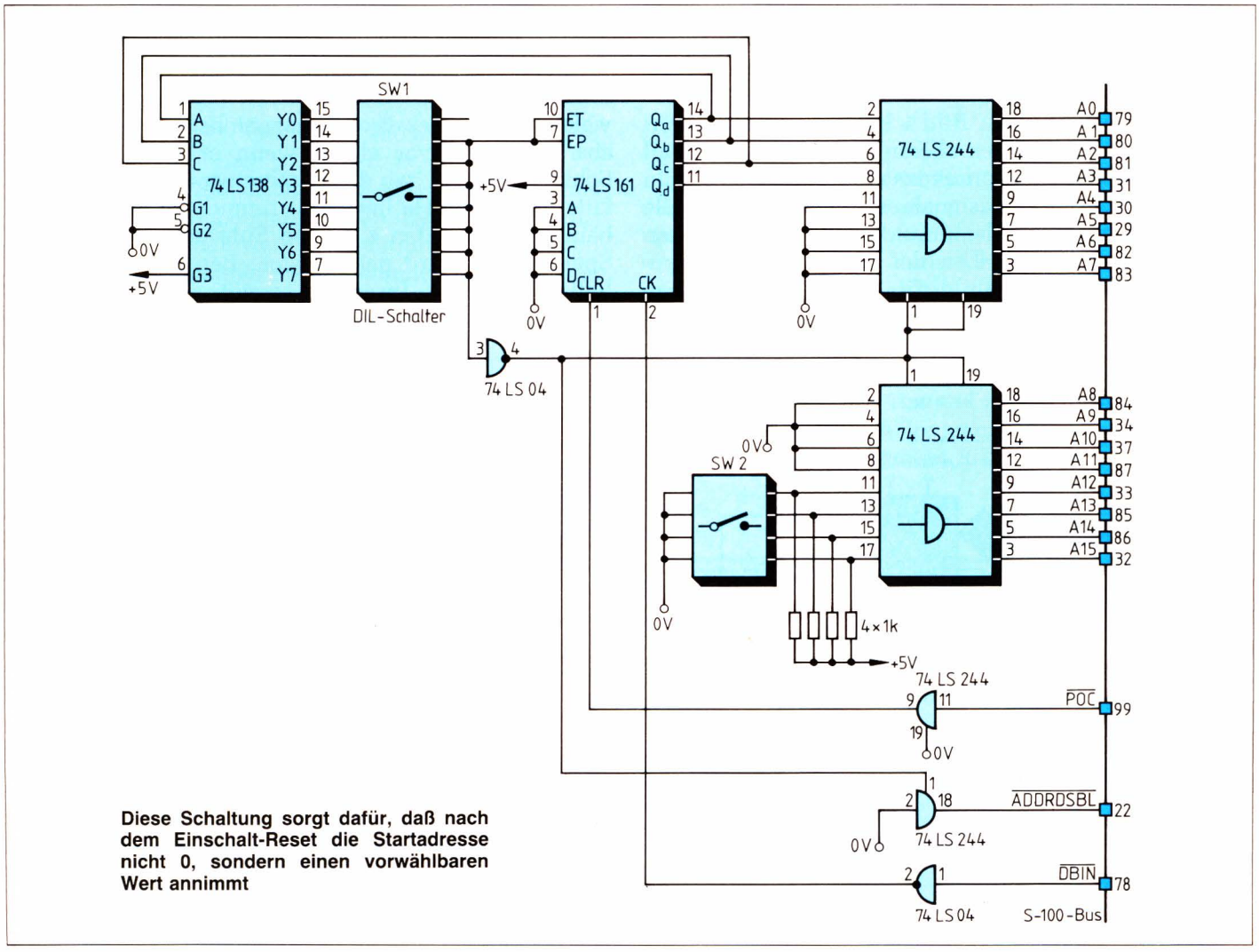
Adressengenerator für den S-100-Bus

Die meisten 8-Bit-Mikroprozessoren lesen nach einem Reset die erste Instruktion bei der Adresse 0. Will man in einem Mikroprozessor-System, z. B. mit S-100-Bus, bei einer anderen Adresse die erste Instruktion lesen, so muß die von der CPU gesendete Adresse transformiert und die gewünschte Adresse von einem Generator in den Bus eingespeist werden. Diese Notwendigkeit ergibt sich, wenn z. B. der RAM-Speicher bei der Adresse 0 anfängt, aber der EPROM-Speicherbereich bei einer von 0 abweichenden Adresse.

Funktionsbeschreibung

Das Signal „POC" (*Power On Clear*), welches auf der CPU-Karte eines S-100-Systems beim Einschalten oder einem manuellen Reset erzeugt wird, setzt die Ausgänge Qa...Qd des Zählers 74LS161 auf Low, wodurch alle Ausgänge (außer dem unbenutzten Y0) des Decodierers 74LS138 auf High gesetzt werden. Dieses H-Potential gelangt über einen DIL-Schalter zum Zähler und verschiedenen Gattern und löst folgende Vorgänge aus:

- Der Zähler wird entriegelt und kann ankommende Clock-Impulse verarbeiten.
- Das Signal „ADDRDSBL" (*Address Disable*) des S-100-Busses geht auf Low und sperrt somit die von der CPU-Karte ausgesendete Adresse.



Diese Schaltung sorgt dafür, daß nach dem Einschalt-Reset die Startadresse nicht 0, sondern einen vorwählbaren Wert annimmt

– Die Puffer 74LS244 werden entriegelt und senden die mit SW2 gewählte Startadresse auf den Bus. Da die Eingänge der Puffer, welche die Adressen A3...A11 steuern, auf Low liegen, können mit SW2 die Adressen zwischen 0 und $F000_H$ in Schritten von 1000_H gewählt werden.

Die CPU erzeugt zur gleichen Zeit ein Lesesignal auf dem Bus, DBIN, d. h. es wird bei der gewählten Adresse gelesen und mit der fallenden Flanke dieses Signals der Zähler um 1 erhöht. Dadurch liegt die für die nächste zu lesende Instruktion erforderliche Adresse auf dem Bus.

Mit den Schaltern SW1 kann man die Anzahl der zu lesenden Instruktionen wählen. Sobald diese Anzahl erreicht ist, wird der Zähler, die Adressenpuffer 74LS244 und das Signal ADDRDSBL verriegelt (Tri-State) und damit der Adressengenerator bis zum nächsten Reset außer Betrieb gesetzt.

Beispiel

Der EPROM-Speicher mit dem BOOT-Programm für die Disketten beginnt mit der Adresse $E000_H$ und enthält in den ersten drei Speicherstellen die Instruktion JUMP START. Für die Mikroprozessoren 8080 und Z80 heißt das in Maschinensprache C3 03 E0. Steht der Schalter SW1 nun in Stellung 3, so liest die CPU diese Sprunganweisung in drei Schritten. Anschließend wird der Adressengenerator außer Betrieb gesetzt, da der Ausgang Y3 des Decodierers 74LS138 auf Low geht, und der eigentliche Programmablauf beginnt bei der Adresse $E003_H$.

Ing. (grad.) Gerhard Trayser

Memory-Mapping für Mikroprozessoren

Mit dieser Schaltung ist es möglich, die Arbeitsweise eines Mikroprozessors optisch zu kontrollieren, indem der gesamte Speicherbereich von z. B. 64 KByte auf einem Oszilloskop dargestellt wird.

Die Schaltung nach *Bild 1* besteht aus zwei D/A-Umsetzern, welche den oberen und den unteren Teil der Adreßleitungen vom Prozessor aufnehmen und in entsprechende Analogsignale umsetzen. Werden alle Adressen nacheinander durchlaufen, ergibt sich ein zeilenweises Beschreiben des Bildschirms entsprechend einem Fernsehbild. Die 256 Zeilen haben im genannten Beispiel 256 Byte Länge. *Bild 2* erläutert den Zusammenhang zwischen der adressierten Speicherzelle und ihrem Platz auf dem Bildschirm. Mit zwei Potentiometern können die Amplituden der Ausgänge auf das Oszilloskop abgestimmt werden, so daß der gesamte Speicherraum genau die Bildschirmfläche einnimmt.

Die optische Darstellung liefert Angaben darüber, welchen Platz des Programmspeichers der Rechner abarbeitet und vor allem, wenn er sich in einer Schleife festläuft, wo diese sich etwa befindet. Ebenfalls deutet die Helligkeit bestimmter Segmente auf häufiges Abarbeiten z. B. von Subroutinen hin. Bei Speichertests wird der getestete Bereich als breiter Balken sichtbar. Insgesamt ergibt sich eine gute Ergänzung zu Software-Testmöglichkeiten, wobei der Anwendungsbereich hauptsächlich bei kleinen Mikrocomputersystemen mit wenigen Prüfmöglichkeiten liegt.

Martin Görlitz

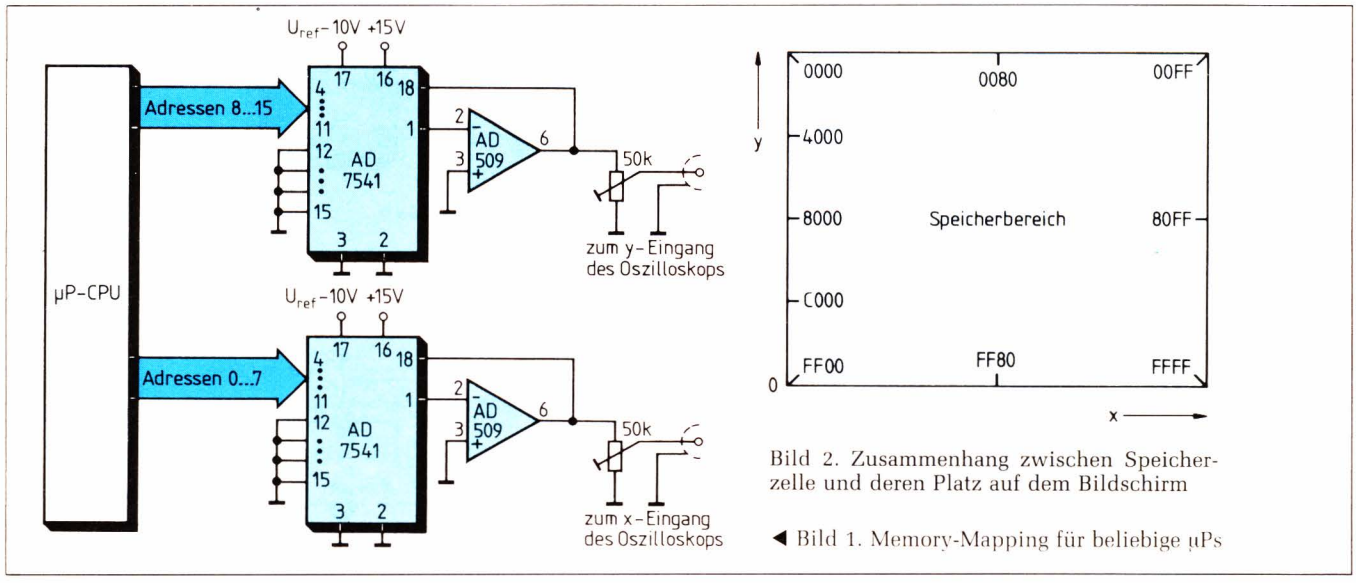


Bild 2. Zusammenhang zwischen Speicherzelle und deren Platz auf dem Bildschirm

◀ Bild 1. Memory-Mapping für beliebige µPs

Erzeugen netzspannungsgesteuerter Programmunterbrechungen

Oft ist es notwendig, einen Mikroprozessor mit einem „Zeitbewußtsein" auszustatten. Das kann durch periodische Programmunterbrechungen geschehen. Für gute Zeitauflösung ist eine hohe Unterbrechungsfrequenz günstig, andererseits sollte im Zeitraum zwischen zwei Unterbrechungen nicht nur die Unterbrechungsbedienung abgeschlossen werden können, sondern auch noch eine ausreichende Reserve zur Abwicklung von Hintergrund-Programmen bleiben. Eine recht gut geeignete und überdies im Mittel ziemlich stabile Unterbrechungsperiode von 10 ms läßt sich aus der Netzfrequenz ableiten.

Programmunterbrechungen zum Zeitpunkt der Nulldurchgänge der Netzspannung bieten zusätzliche Vorteile:

- Triacs und Thyristoren können ohne großen Aufwand im Nulldurchgang der Netzspannung getriggert werden.
- Schrittmotor-Steuerungen lassen sich sehr preiswert verwirklichen, wenn man die Schritt-Impulse mit den Netzhalbwellen synchronisiert. Man kann dann die Endstufen mit pulsierendem Gleichstrom speisen.
- Oft werden zur Erfassung von Analog-Signalen schnelle Analog/Digital-Umsetzer mit vorgeschalteten Multiplexern verwendet. Wenn während jeder Netzhalbwelle alle Analogsignale digitalisiert werden und die Abtast-Zeitpunkte gegenüber den Nulldurchgängen der Netzspannung konstant verzögert sind, kann man durch Bildung des arithmetischen Mittels zweier aufeinanderfolgender Meßwerte desselben Kanals den Einfluß eingestreuter 50-Hz-Brummspannungen weitgehend reduzieren.

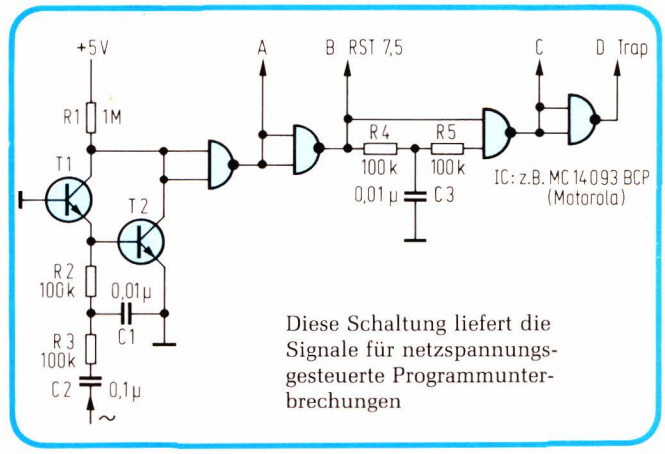
Netzspannungsgesteuerte Programmunterbrechungen sind aber auch von Bedeutung, wenn der µP auf einen Netzspannungsausfall aufmerksam gemacht werden muß. Geschieht dies rechtzeitig, so kann er noch Notmaßnahmen treffen, z. B.:

- wichtige Daten in einem nichtflüchtigen Speicher retten,
- ein Notsignal an eine übergeordnete Steuerung absenden,
- eine Notstromversorgung einschalten.

Die im *Bild* dargestellte Schaltung erzeugt Unterbrechungssignale, die für solche Zwecke benutzt werden können. Der aus C 2, R 3, C 1 und R 2 bestehende Bandpaß dient zur Unterdrückung von hochfrequenten Einstreuungen und Störimpulsen. Während der positiven Halbwellen des Bandpaß-Ausgangsstroms i ist Transistor T 2 gesättigt, während der negativen Halbwellen T 1. Nur während kurzer Zeitintervalle um die Nulldurchgänge von i entstehen an den Kollektoren von T 1 und T 2 kurze positive Impulse. Die Phase von i und damit die zeitliche Lage dieser Impulse bezüglich der Nulldurchgänge der Netzspannung kann man durch geeignete Wahl von C 2 verändern. Die nachgeschalteten Schmitt-Trigger versteilern die Impulsflanken und komplementieren das Unterbrechungssignal. Die positiven Impulse am Ausgang B eignen sich z. B. für den (maskierbaren) Unterbrechungs-Eingang RST 7.5 des Mikroprozessors 8085. Die Dauer der Ausgangsimpulse nimmt mit steigender Amplitude der Eingangsspannung ab. Bei einer Eingangsspannung von U_{eff} = 10 V beträgt sie 0,3 ms.

Im Normalbetrieb verhindert der Tiefpaß R 4, C 3, daß die Nulldurchgangsimpulse zu den Ausgängen C und D gelangen. Erst wenn die Netzspannung für einige ms ausgefallen ist, ändern sich die komplementären Zustände dieser Ausgänge. Das positive Alarmsignal am Ausgang D eignet sich z. B. für den (nichtmaskierbaren) Unterbrechungs-Eingang TRAP des Typs 8085. Die zum Erkennen eines Netzausfalls erforderliche Zeit hängt von der z. B. durch C 1 beeinflußbaren oberen Grenzfrequenz des Eingangsfilters und von der Zeitkonstanten R 4 · C 3 ab.

Dr. Martin Brendle



Diese Schaltung liefert die Signale für netzspannungsgesteuerte Programmunterbrechungen

Automatisches Umschalten eines Druckers zwischen mehreren Rechnern

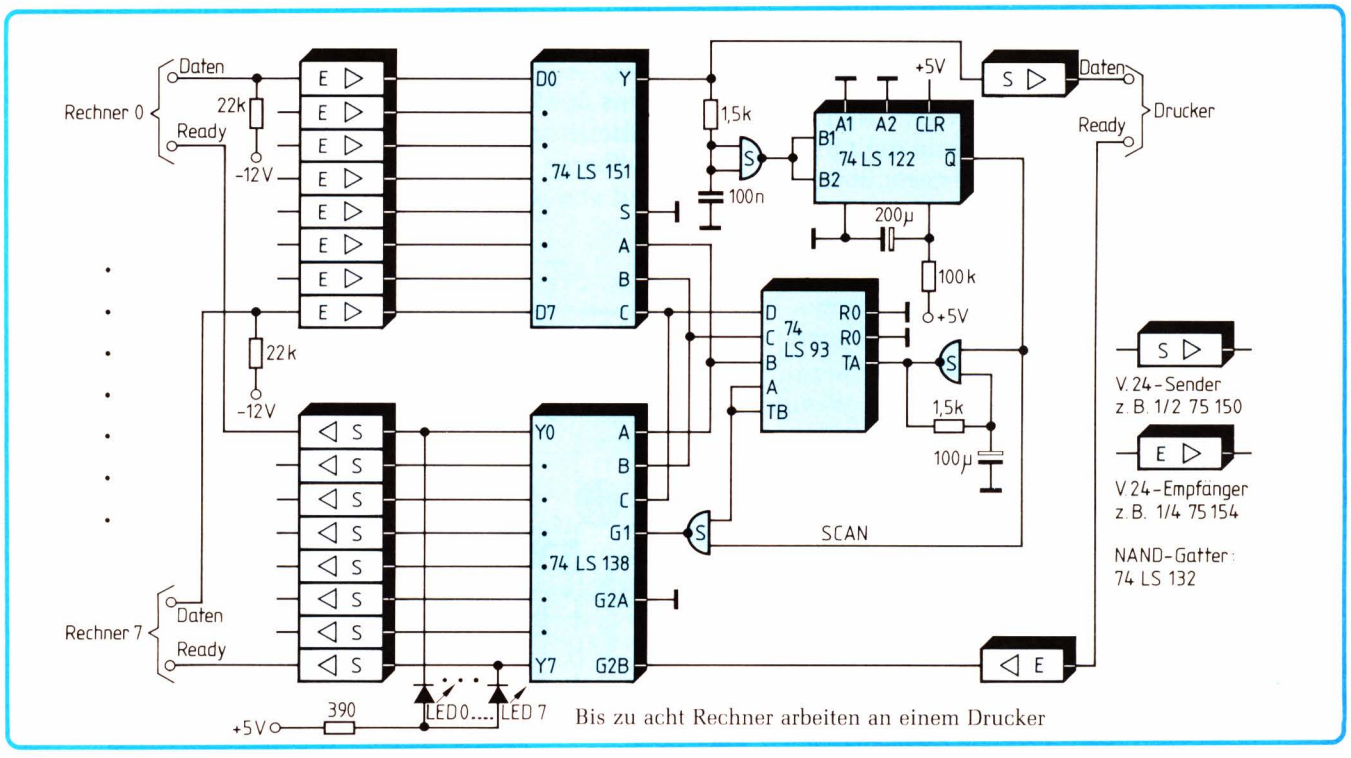
Häufig steht für mehrere Kleinrechner oder Mikrocomputer-Entwicklungssysteme nur ein gemeinsamer Drucker zur Verfügung. Das erforderliche Umschalten oder -stecken ist ein Quelle steten Ärgers. Die beschriebene Anordnung führt das Umschalten zwischen bis zu acht Rechnern automatisch durch.

Die Schaltung ist anwendbar, wenn
- die Daten zum Drucker über eine serielle, asynchrone V.24-Leitung übertragen werden und
- der Drucker Bereitschaft durch einen „H-Pegel" auf einer V.24-Leitung (Ready, Data Terminal Ready) zurückmeldet.

Ein 8 : 1-Multiplexer 74LS151 führt die acht von den Rechnern kommenden Datenleitungen an den Drucker. Der Demultiplexer 74LS138 legt an eine der acht zu den Rechnern führenden Leitungen ein Ready-Signal. Multiplexer und Demultiplexer werden von einem 4-Bit-Zähler gesteuert. Solange keiner der Rechner Daten zum Drucker sendet, ist die Leitung ‚SCAN' auf „H"-Pegel, und ein Multivibrator taktet den Zähler mit etwa 4 Hz. Dadurch legt der Demultiplexer zyklisch nacheinander an alle acht Rechner ein Ready-Signal; dieser Vorgang wird von den LEDs 0...7 angezeigt. Ein Rechner, der Daten zum Drucker senden will, darf damit erst beginnen, wenn er das Ready-Signal erhält. Der dann zum Drucker übermittelte Datenstrom triggert das Monoflop 74LS122, ‚SCAN' geht auf „L"-Pegel, der Taktgenerator wird gestoppt, und der Zähler bleibt in der momentanen Position stehen. Wenn in der Datenübertragung eine Pause von mehr als 10 s eintritt, geht das Monoflop in die Ruhelage zurück und der Suchvorgang beginnt wieder. Alle Ein- und Ausgänge der Schaltung müssen über V.24-Sender bzw. -Empfänger an die Leitungspegel angepaßt werden.

Das automatische Umschalten ist für die Rechner unsichtbar, so daß keine Software-Änderungen erforderlich sind. Es muß lediglich sichergestellt werden, daß während eines Druckvorgangs keine Pause von mehr als 10 s Dauer auftritt. – Ein Mustergerät verbindet drei Prozeßrechner und drei Mikrocomputer-Entwicklungsplätze mit einem Kettendrucker und wird von allen Benutzern als große Erleichterung empfunden.

Dipl.-Ing. Nicholas Roethe



Bis zu acht Rechner arbeiten an einem Drucker

Der Schutz von Mikrocomputern gegen elektrische Störungen

Zahlreiche Mikrocomputer-Installationen in Fertigungseinrichtungen bzw. Maschinensteuerungen erfahren oft Schwierigkeiten durch elektrische Störungen – meist erst erkannt nach dem Einbau. Die folgenden Anmerkungen sollen hier schon vorab eine kleine Hilfestellung bieten.

1 Ursache von elektrischen Störungen

Zu den auslösenden Faktoren muß hier zuerst der hohe Stromverbrauch der Steuerungsanlagen gerechnet werden, aber auch Umweltbedingungen, wie z. B. Gewitter, können die Ursache sein.

Um Störeinflüsse zu vermindern, müssen oft nur die Installations-Anweisungen der Rechner- und Instrumentenhersteller und die Verdrahtungsnormen (VDE-Vorschriften) innerhalb des Betriebes sorgfältig beachtet werden. Folgende physikalische Einflüsse lösen Störungen aus:

- Beeinflussungen durch die Netzfrequenz und deren Oberwellen:
  Induktive Kopplung
  Kapazitive Kopplung
  Widerstandskopplung

- Elektromagnetische Störungen entstehen durch:
  Leitungen mit hohen Strömen
  Schweißmaschinen
  Transformatoren

- Elektrostatische Störungen entstehen durch kapazitive Kopplungen mit:
  Leitungen
  Relais
  Schaltern

- Hf-Störungen entstehen durch:
  Funken an Bürsten von Elektromotoren
  Lichtbögen (Schalter, Relais usw.)
  Korona bei Hochspannungsleitungen

2 Möglichkeiten der Störungs-Unterdrückung

2.1 Bei Signalleitungen

- Serielle Signalübermittlung: Für die Übertragung von Analogsignalen werden verdrillte Kabel verwendet, die, wenn es notwendig ist, auch abgeschirmt sein können (*Bild 1*).

  Oft empfiehlt es sich, Analogsignale zu digitalisieren und diese als wenig störanfällige digitale Informationen zu übertragen. Vorzugsweise werden digitale Schaltsignale mit 24, 48, 115 und 220 V usw. übertragen. Um erhöhte Kabelausgaben zu vermeiden, empfiehlt es sich, die optimale Übertragungsspannung und die benötigte Abschirmung im Betrieb zu erproben. Oft ist aber die Betriebsspannung durch die Schaltelemente und Vorschriften bestimmt.

- Parallele Signalübertragung: Bei Längen von mehr als 100 m wird die parallele Datenübertragung unwirtschaftlich und zu störanfällig. Für kurze Entfernungen genügt hingegen eine parallele Datenübertragung mit einem Flachbandkabel, bei dem

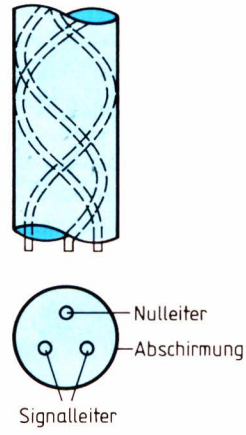


Bild 1. Das verdrillte Kabel mit Abschirmung

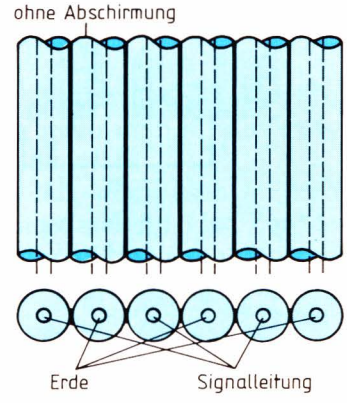


Bild 2. Störschutz beim Flachbandkabel durch Erden jedes zweiten Leiters

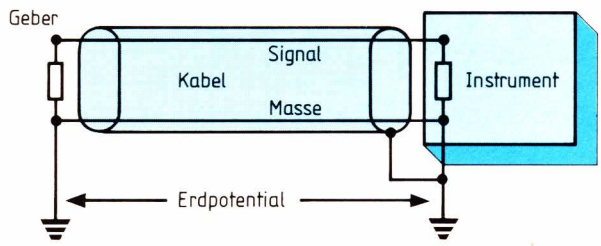


Bild 3. Beidseitiges Erden einer Signalleitung provoziert – durch unterschiedliche Erdpotentiale an den Leitungsenden – oft Störungen

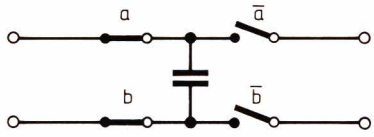


Bild 4. Potentialtrennung durch eine „fliegende Kapazität"

jeder zweite Leiter zur Abschirmung geerdet ist (*Bild 2*).

Wegen der großen Störsicherheit wird man jedoch bevorzugt die für alle genormten Spannungen erhältlichen Parallel/Seriell-Umsetzer einbauen und mit abgeschirmten verdrillten Kabeln verbinden. Für besonders störungsfreie Übertragung wird das koaxiale Kabel empfohlen.

2.2 Potentialtrennung

Da das Erdpotential an verschiedenen Stellen meistens unterschiedlich ist, sollten Masseleitungen nur an einem Ende geerdet und somit ein unerwünschter Stromdurchfluß in der Geberleitung oder Abschirmung verhindert werden (*Bild 3*). Wird z. B. die Abschirmung an zwei Stellen geerdet (ein häufiger Fehler), so kann sie demzufolge Fremdsignale auf die Signalleitungen einstreuen. Potentialströme in der Geberleitung können durch Potentialtrennung beseitigt werden. Man verwendet dazu folgende Prinzipien:

- „Fliegende Kapazitäten" (*Bild 4*)
- Optokoppler
- Relais
- Transformatoren

Die Optokoppler sind hierbei sicher die heute am weitesten verbreiteten Elemente.

2.3 Filter

Störsignale, die in Geberschaltkreisen sind, lassen sich fast immer durch Filter beseitigen. Besonders häufig treten 50-Hz-Störungen auf, die von der Stromversorgung nicht abgeblockt wurden. Störungen, die dieselbe Frequenz wie das Signal haben, verursachen die größten Schwierigkeiten. Sie müssen direkt an der Störquelle (beispielsweise durch ein RC-Glied) unterdrückt werden.

2.4 Erden von Geräten

Potentialströme werden immer entstehen, wenn Rechner mit der vorhandenen Peripherie an eine gemeinsame Erdleitung seriell angeschlossen werden. Durch eine radiale Anordnung der Erdleitungen lassen sich diese Erdpotentialströme aufheben (*Bilder 5* und *6*).

2.5 Trennung der Signal-Übertragungssysteme

Weiter kann man besondere Signalübertragungssysteme für das Erkennen und Übertragen von Signalen einsetzen. Dies wird man immer dann tun, wenn Maschinen mit hoher elektrischer Störanfälligkeit und komplexen Schaltungen einzusetzen sind. In diesem Zusammenhang soll natürlich die besonders störungsarme Übertragung über Lichtleiter nicht vergessen werden. Gerade diese Möglichkeit wird man sicher in den nächsten Jahren vermehrt realisieren.

Dipl.-Ing. Peter Klis

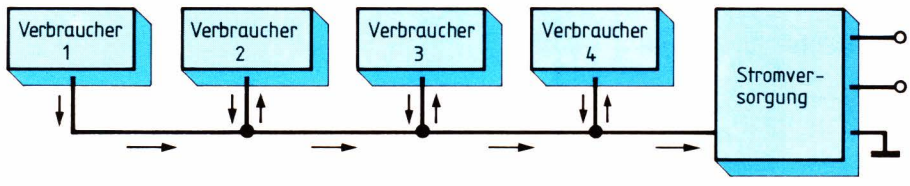


Bild 5. Serielle Erdung mehrerer Verbraucher; eine störanfällige Methode

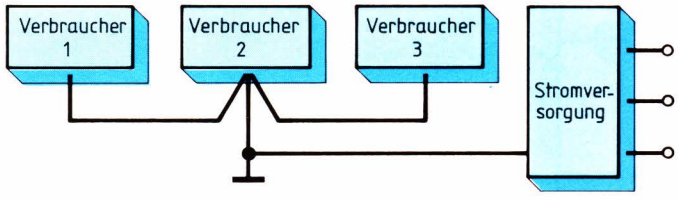


Bild 6. Durch „radiale" Erdung lassen sich zahlreiche Störungen von vorneherein eliminieren

Wurzelziehen durch sukzessive Approximation

Will man aus dem Leistungsdichtespektrum das Amplitudenspektrum berechnen, dann kommt man um das Wurzelziehen nicht herum. Der im folgenden beschriebene Algorithmus erfordert weder „echte" Division noch Multiplikation.

Ein bekanntes Wurzelziehverfahren [1] benutzt die numerische Reihe:

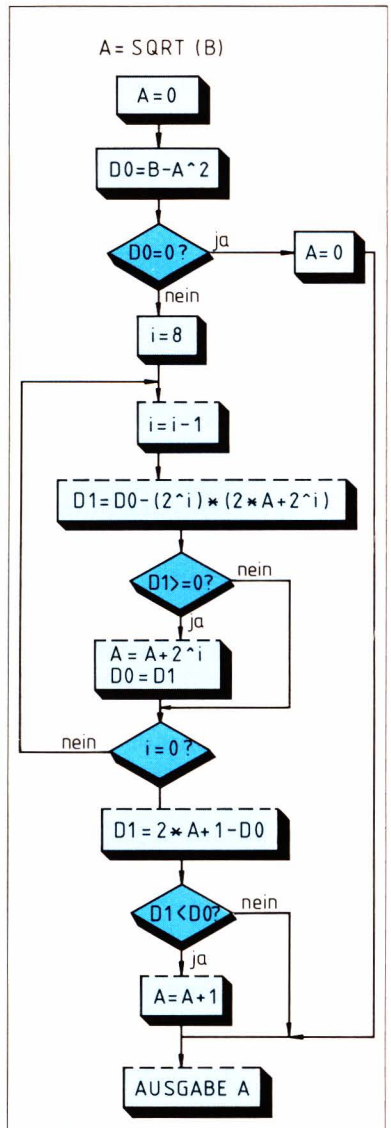
$$1+3+5+..+(2n-1) = n^2 \qquad (1)$$

Es werden normalerweise (wenn nicht mehrere „Startwerte" vorgegeben werden) I=n Iterationsschritte benötigt. Das Verfahren hat den Vorteil, daß nur Addition und Subtraktion benötigt werden.

Dann gibt es noch das Newtonsche Verfahren, das im Spezialfall des Wurzelziehens A=SQRT(B) auf die rekursive Formel

$$A(n+1) = (A(n) + B/(A(n))/2 \qquad (2)$$

führt. Bei Vorgabe der „nächstgelegenen" Zweierpotenz als Startwert A(0) ist bereits nach einem einzigen Iterationsschritt der relative Fehler nur noch 25 %. Dem Vorteil der schnellen Konvergenz steht aber leider der Nachteil gegenüber, daß man spätestens beim zweiten Iterationsschritt nicht mehr ohne echte Multiplikation bzw. Division auskommt.



Mit Addition, Subtraktion und Schiebebefehlen kommt dieser Wurzelalgorithmus aus

Der hier vorgestellte Algorithmus zieht aus einem 16-Bit-Wert die Wurzel. Durch Anwendung der sukzessiven Approximation läßt sich die Anzahl der Iterationsschritte auf I = ld(n) verkürzen. Im „Programm" wird mit n = 256 Möglichkeiten I = 8 vorgegeben.

Das Flußdiagramm (*Bild*) zeigt den Ablauf der Operation. Für A=SQRT(B) muß B im Bereich $0 \leqq B \leqq 65535$ vorgegeben werden. A wird mit 0 vorbelegt, und D0 wird ein Wort zugewiesen:

$$D0 = B - A \wedge 2 \qquad (3)$$

Wenn D0 = 0 ist, kann A = 0 als Ergebnis ausgegeben werden.

Die sukzessive Approximation beginnt mit dem MSB von A (i=8): Durch die Zuweisung D0 = D1 wird im nächsten Iterationsschritt (i=7) das Quadrieren von A gespart. Der Term $(2 \wedge i)*(2*A+2 \wedge i)$ kann im Programm vollständig mit Schiebeschritten und einer Addition ausgeführt werden. Der Schleifenabbruch ist durch i=0 gegeben.

Nach der Iterationsschleife folgt noch der Test, ob das letzte Bit um 1 erhöht werden muß. Es gilt:

$$SQRT(B)-1 < A \leqq SQRT(B) \qquad (4)$$

und deswegen

$$SQRT(B) < A+1 \leqq SQRT(B)+1$$

Also muß die Abweichung

$$D1 = (A+1) \wedge 2 - B \qquad (5)$$

mit D0 (siehe Gleichung (3)) verglichen werden. Für (5) läßt sich mit (3) auch

$$D1 = 2*A+1-D0 \qquad (6)$$

schreiben. Das heißt, man kommt wieder ohne Multiplikation aus. Ist D1–D0<0, dann muß für das endgültige Ergebnis A inkrementiert werden.

Aus dem Algorithmus läßt sich unschwer erkennen, daß der absolute Fehler maximal ±0,5 ist. Zur Messung der Rechenzeit wurden in PLM-86 die Wurzeln von 1...50 000 gezogen. Dazu wurden 29 s benötigt. Damit liegt die Ausführungszeit unter 600 µs (Taktfrequenz des 8086: 5 MHz).

Dipl.-Ing. Kurt Enzmann

Literatur

[1] *Bronstein, Semendajew*: Taschenbuch der Mathematik (S. 137). Verlag Harri Deutsch), Zürich, Frankfurt, Thun.

[2] *Rix, P.*: Universeller Quadratwurzel-Algorithmus. ELEKTRONIK 1982, H. 23, S. 81...82

Zahlenumwandlung in PASCAL

Bei Programmen, die die Programmierung von PROMs, EPROMs und PALs unterstützen, benötigt man z. B. bei Adreß- und Datendarstellungen Zahlkonvertierungsprogramme.

Bei den höheren Programmiersprachen wird üblicherweise bei den Typen INTEGER und REAL das dezimale Zahlsystem verwendet. Für die oben genannten Aufgabenstellungen braucht man jedoch u.a. die duale und die sedezimale Darstellung.

```
program dezson_test;
(*COPYRIGHT (c) 1984 by Peter Warlies, 7800 Freiburg*)

const
  nl=50;
  bmax=36;
type
  zeiket= string[nl];
(*-------------------------------------------------------------*)
function  dezson(  k: integer;  (*eingabe: dez. zahl *)
                   b: integer   (*eingabe: basis zielsystem*)
                              ): zeiket;
(*
  dezson wandelt die dezimale zahl k in eine
  zeichenkette in dezson um;
  b ist die basis des zielsystems
*)
var
  jd,h,r,l      : integer;
  a,son,ve      : zeiket;
(*
division mit rest;
h: zwischenergebnis
r: rest
l: ziffernstellen im zielsystem (= nl
*)

begin
  ve:='0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';
  if(b (= bmax) then
    begin
      a:='';  r:=1;  h:=1;  l:=1;
      while ((r)0) or (h)0)) and (l(=nl) do
        begin
          h:=trunc(k/b);  r:=k-h*b;  l:=l+1;
          if h)= 0 then
            begin
              a:= a+copy(ve,r+1,1);  k:=h;
            end;
        end; (*while r)0 *)
      son:='';
      for jd:= length(a) downto 1 do
        son:= son + copy(a,jd,1);
     'dezson:= son;
    end
  else
    begin
      writeln;  writeln('ERROR fkt dezson: zielbasis ) ', bmax);
      dezson:='';
    end;
end;(*function dezson *)
(*-------------------------------------------------------------*)
var
  i: integer;
  b,c,d,e,f,g,o,h: zeiket;

begin
  writeln('Dezimal-Basis-Konvertierung');
  writeln;
  writeln('basis:  10          2          3    4    5    6    7    8   16');
  writeln('*****************************************************************');
  for i:= 1 to 19  do
    begin
      b:=dezson(i,2); c:=dezson(i,3); d:=dezson(i,4); e:=dezson(i,5);
      f:=dezson(i,6); g:=dezson(i,7); o:=dezson(i,8); h:=dezson(i,16);
      writeln( i:10, b:10, c:10, d:5, e:5, f:5, g:5, o:5, h:5);
    end;
end.
```

```
basis:  10          2          3    4    5    6    7    8   16
*****************************************************************
         1         01         01   01   01   01   01   01   01
         2        010         02   02   02   02   02   02   02
         3        011         010  03   03   03   03   03   03
         4       0100         011  010  04   04   04   04   04
         5       0101         012  011  010  05   05   05   05
         6       0110         020  012  011  010  06   06   06
         7       0111         021  013  012  011  010  07   07
         8      01000         022  020  013  012  011  010  08
         9      01001        0100  021  014  013  012  011  09
        10      01010        0101  022  020  014  013  012  0A
        11      01011        0102  023  021  015  014  013  0B
        12      01100        0110  030  022  020  015  014  0C
        13      01101        0111  031  023  021  016  015  0D
        14      01110        0112  032  024  022  020  016  0E
        15      01111        0120  033  030  023  021  017  0F
        16     010000        0121 0100  031  024  022  020  010
        17     010001        0122 0101  032  025  023  021  011
        18     010010        0200 0102  033  030  024  022  012
        19     010011        0201 0103  034  031  025  023  013
```

Bild 2. Einige Ergebnisse als Beispiel

Bild 1. Dieses PASCAL-Programm sorgt für die Umsetzung dezimaler Zahlen in solche beliebiger Basis (b)

Bild 1 zeigt das Listing eines Unterprogramms „dezson" in PASCAL, das die Konvertierung aus dem Quellzahlsystem mit der Basis 10 in ein Zielsystem mit der Basis b erlaubt.

Aus dem mitabgedruckten Hauptprogramm und dem Ergebnisausdruck in *Bild 2* kann man einige Aufruf- und Anwendungsmöglichkeiten des Funktionenunterprogramms erkennen. Das Programm „dezson" benötigt als Eingabeparameter die zu konvertierende dezimale Zahl k im Quellsystem und die Basis b des Zielsystems. Als Ergebnis liefert das Programm eine Zeichenkette in der Variablen dezson (Aufrufname der Funktion).

Der Algorithmus basiert auf der Division mit Rest. Nach einer Konvertierung mit dem Programm nach Bild 1 kann man mit geringen Programmzusätzen noch das Intel-HEX-, das BHLF- und das JEDEC-Format für die Programmiergeräte erzeugen.

Ein Programm zur Rückwandlung aus dem Zielsystem mit der Basis b in das Quellsystem mit der Basis 10 und eine Realisierung in anderen höheren Programmiersprachen, wie z. B. BASIC, ist leicht möglich. *Peter Warlies*

Grafikprozessor in MPL programmiert

Der Grafikprozessor EF9366 wird üblicherweise in Assemblersprache programmiert. Das kann eine sehr langwierige Angelegenheit sein. Software für diesen Baustein läßt sich schneller entwickeln und testen, wenn man eine höhere Programmiersprache verwendet. Besonders gut geeignet ist MPL, eine PL/1-Abart, die speziell an diese Aufgabe angepaßt wurde.

Der MPL-Compiler wurde für die Prozessoren 6800 und 6809 entwickelt. Er erzeugt Assemblercode, der zusammen mit Modulen aus einer MPL-Bibliothek und externen Modulen zu einem Gesamtprogramm verbunden werden kann. Das Quellprogramm schreibt man mit Hilfe eines Texteditors. Im Vergleich zur Assemblersprache bietet MPL wesentlich komfortablere Möglichkeiten: Die Sprache läßt Variablenmanipulation, arithmetische Operationen und strukturierte Programmierung zu, ohne die Maschinennähe aufzugeben. Einige Eigenschaften von MPL:

Variablentypen:
- binär, ein oder zwei Bytes, mit oder ohne Vorzeichen
- bitbezogen
- dezimal
- Feld, max. dreidimensional;

Arithmetische Operationen:
- Addition
- Subtraktion
- Multiplikation
- Division;

Logische Operatoren:
- Shift
- AND
- OR;

Steuerstrukturen:
- IF THEN ELSE
- DO WHILE
- DO TO (FOR-NEXT-Schleife)
- Prozeduraufruf mit Parameterübergabe;

Register- und Variablen-Deklaration

In einem MPL-Programm müssen die Register des EF9366 deklariert werden. Dabei kann man die verschiedenen Variablentypen auch mischen (z. B. binäre mit Bitvariablen). Nicht verwendete Bits einzelner Register muß man nicht vereinbaren. So wird im Deklarationsteil, der in *Bild 1* gezeigt ist, READY als Bitvariable definiert, die sich auf Bit 2 des Statusregisters bezieht. Dieses Bit zeigt an, ob der Grafikprozessor einen Befehl empfangen kann oder nicht. Die weiteren Variablenvereinbarungen in Bild 1 beziehen sich auf die Register CSIZE und CTRL2 sowie auf ein gesondert definiertes Farbregister (nicht im Grafikprozessor). Die Farbvariablen repräsentieren in diesem Fall konstante Werte. Im einzelnen bedeuten:

BIN eine vorzeichenlose 1-Byte-Variable
BIN (2) eine vorzeichenlose 2-Byte-Variable
BIT (n) eine n-Bit-Variable

Die n-Bit-Variable dient dazu, ein n-Bit-Feld im spezifizierten Byte festzulegen. Die Pseudobefehle DEF und CONST definieren entweder eine physikalische Adresse oder einen symbolischen Wert.

Prozeduren

Unterprogramme werden in MPL als Prozeduren formuliert. Die Parameter kann man dabei auf zwei verschiedene Arten übergeben. In der einen Form übergibt man eine beliebig lange Parameterliste, die in runden Klammern eingeschlossen ist:

```
/* ************************************************/
/*                                                */
/*          EF 9366  DECLARATIONS                 */
/*                                                */
/* ************************************************/

  DECLARE 1 GDP DEF $F820,                  ! F820=GDP address
            2 STATUS,                       ! status register
                3 NUS1  BIT(5),             ! not used
                3 READY BIT(1),             ! GDP ready
                3 NUS2  BIT(2),             ! not used
            2 COMAND BIN  DEF STATUS,       ! command register
                                            ! same address as STATUS
            2 CTRL1,                        ! CTRL1 register
                3 NUS3  BIT(6),             ! not used
                3 PENER BIT(1),             ! pen,eraser selection
                3 ONOFF BIT(1),             ! up,down
            2 CTRL2 BIN,                    ! CTRL2 register
            2 CSIZE BIN,                    ! CSIZE register
            2 NUS4,                         ! byte not used
            2 DELTAX BIN,                   ! DELTAX register
            2 NUS5,                         ! byte not used
            2 DELTAY BIN,                   ! DELTAY register
            2 X366 BIN(2),                  ! X register
            2 Y366 BIN(2)                   ! Y register

/***************************************/
/*                                     */
/*    EXTERNAL REGISTERS AND           */
/*    GLOBAL VARIABLES                 */
/*                                     */
/***************************************/
```

```
DECLARE COLOR DEF $F831          ! color register
/*     GLOBAL VARIABLES                 */

/*     variables for CSIZE register  */

 DCL       ZERO       CONST(0),
           ONE        CONST(1),
           TWO        CONST(2),
           THREE      CONST(3),
           FOUR       CONST(4),
           FIVE       CONST(5),
           SIX        CONST(6),
           SEVEN      CONST(7),
           EIGHT      CONST(8),
           NINE       CONST(9),
           TEN        CONST($A),
           ELEVEN     CONST($B),
           TWELVE     CONST($C),
           THIRT      CONST($D),
           FOURT      CONST($E),
           FIFT       CONST($F),
           SIXT       CONST($0)

/*   variables for color register    */

 DCL       WHITE      CONST(0),
           CYAN       CONST(1),
           MAGEN      CONST(2),
           BLUE       CONST(3),
           YELLOW     CONST(4),
           GREEN      CONST(5),
           RED        CONST(6),
           BLACK      CONST(7)

/*   variables for  CTRL2  register  */

 DCL       VERTI      CONST(1),
           HORIZ      CONST(0),
           SLANT      CONST(1),
           NORMA      CONST(0),
           CONTI      CONST(0),
           DOTTED     CONST(1),
           DASHED     CONST(2),
           MIXED      CONST(3)
```

Bild 1. Deklarationsteil eines MPL-Programms: Unterschiedliche Variablentypen können auch gemischt verwendet werden, z. B. die Typen BIT und BIN. Die Vereinbarungen beziehen sich auf die Routinen in Bild 2 und 3

CALL Prozedurname (p1, p2, ..., pn)

In der zweiten Form ist die Anzahl der Parameter auf drei beschränkt, sie werden durch die Akkumulatoren A und B sowie das X-Register des Mikroprozessors übergeben:

CALL Prozedurname <p1, p2, p3>

Bild 2 zeigt zwei Beispiele von Prozeduren. Die erste, in der die Skalierungsfaktoren X und Y des CSIZE-Registers gesetzt werden, läßt sich mit dem Befehl

CALL SETSIZ <TWO, TWO,>

aufrufen. Dabei wird das Register mit dem Bitmuster 00100010 geladen. Auf ähnliche Weise verändert man den Inhalt des Registers CTRL2, das die Lage der Buchstaben und den Linientyp festhält. Der Aufruf kann z. B. so aussehen:

CALL SETCR2 (HORIZ, NORMA, CONTI)

Damit legt man die Zeichen auf „Nicht-Italic" in horizontaler Richtung fest, und die Linien sind durchgehend. Die unteren vier Bits sind bei dieser Betriebsart 0.

Bild 3 zeigt ein Beispiel für ein MPL-Hauptprogramm, das den Vereinbarungsteil von Bild 1 und die Prozeduren von Bild 2 verwendet. Außerdem werden zwei weitere Prozeduren, DELAY und WRITE, aufgerufen, die nicht im Bild dargestellt sind. Das Programm initialisiert den Grafikprozessor und löscht den Bildschirm. Nachdem der Prozessor bereit ist, einen Befehl entgegenzunehmen (READY = 1), setzt es die Skalierungsfaktoren X und Y und initialisiert den Textpuffer MESS. Danach wird der Text in einer Schleife abwechselnd auf den Bildschirm geschrieben und wieder gelöscht.

Rudolf Hofer

Literatur

[1] Programming the EF9365/EF9366 graphic display processor using the MPL language. Application Note NA-023A, Thomson-Efcis.


```
/* ************************************** */
/*                                        */
/*   PROCEDURE SETSIZ:                    */
/*   SET CSIZE REGISTER                   */
/*      CALLING: CALL SETSIZ<P1,P2,>      */
/*               P1: X SCALE              */
/*               P2: Y SCALE              */
/*                                        */
/* ************************************** */

SETSIZ:
         PROC<ECHX,ECHY,>

 DECLARE ECHX BIN,        ! for parameters
         ECHY BIN


 ECHX=ECHX SHIFT 4        ! ECHX shifted in the 4 MSB
 CSIZE=ECHX IOR ECHY      ! write in CSIZE

RETURN
END SETSIZ


/* ********************************** */
/*                                    */
/*   PROCEDURE SETCR2                 */
/*   SET CNTRL2                       */
/*    CALLING:CALL SETCR2(P1,P2,P3)   */
/*            P1: CHARACTERS          */
/*                ORIENTATION         */
/*            P2: CHARACTERS TYPE     */
/*            P3: TYPE OF VECTORS     */
/*                                    */
/* ********************************** */

SETCR2:
      PROC(PR1,PR2,PR3)

 DECLARE PR1 BIN,        ! for parameters
         PR2 BIN,
         PR3 BIN


PR1=PR1 SHIFT 3      ! justifies parameters in byte
PR2=PR2 SHIFT 2

CTRL2=PR1 IOR PR2 IOR PR3

RETURN
END SETCR2
```

Bild 2. MPL-Prozeduren zum Setzen der Register CSIZE und CTRL2

```
/*****************************************/
/*                                       */
/*  WRITE AND FLICKER OF A TEXT          */
/*                                       */
/*****************************************/

EX1:
     PROCEDURE OPTIONS(MAIN)

 DECLARE MESS CHAR(30),   ! text buffer
         WAIT BIN(2)      ! for delay

/*   GDP initialization      */

      COMAND=$6           ! clear screen

            DO WHILE READY=0
            END           ! GDP ready ?

      CTRL1=$0            ! inhibition of interruptions
      ONOFF=1             ! pen down


  COLOR=RED
  CALL SETSIZ<TWO,TWO,>

MESS='THOMSON-EFCIS'     ! initialization of text

/*   writing loop       */

LOOP:
          PENER=1      ! pen selection
          X366=100     ! initialize X and Y
          Y366=20
                CALL WRITE<13,,>  ! write 13 characters

             WAIT=20000
                CALL DELAY(WAIT)  ! delay

          PENER=0       ! selection of eraser
          X366=100
          Y366=20
                CALL WRITE<13,,>

             WAIT=10000
               CALL DELAY(WAIT)

 GOTO LOOP
END EX1
```

Bild 3. Beispiel eines MPL-Hauptprogramms, das einen blinkenden Text auf dem Bildschirm darstellt

BASIC-Programm simuliert PLL-Schaltung

Beim Entwurf von PLL-Schaltungen tritt häufig das Problem auf, daß sich das Einschwingverhalten des Ausgangssignales nur mit komplizierten mathematischen Beziehungen bestimmen läßt. Um einem Entwickler den schnellen Überblick über das Verhalten seiner Schaltung zu geben, schrieb der Autor dieses Beitrages ein BASIC-Programm, das die Ausgangsfunktion direkt auf dem Bildschirm darstellt und bei dem die Auswirkungen von Parameteränderungen direkt sichtbar sind. In diesem Programm kommen nur konventionelle mathematische Methoden zur Anwendung.

Zunächst sei die Funktionsweise einer typischen PLL-Schaltung erläutert, soweit sie für das Verständnis des Programms erforderlich ist (*Bild 1*): Die Frequenz des Ausgangssignals einer VCO-Schaltung ist mit Hilfe folgender Beziehung zu bestimmen:

$$f_{VCO} = k_o \cdot u_g + f_o \qquad (1)$$

f_o: Freilauffrequenz ($U_g = 0$);

u_g: Steuerspannung des VCO;

k_o: Konstante zur Umrechnung der Steuerspannung in Oszillatorfrequenz (Hz/V).

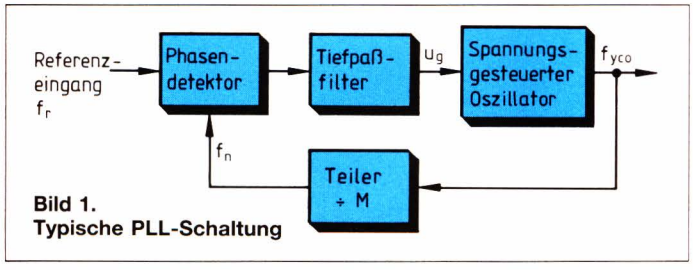


Bild 1. Typische PLL-Schaltung

Vorausgesetzt wird dabei ein lineares Verhalten des VCO, zumindestens in dem Bereich, über den die Oszillatorfrequenz verändert werden soll. Häufig teilt man die Ausgangsfrequenz des Oszillators durch den Faktor n, bevor sie mit einer gegebenen Referenz verglichen wird. In diesem Fall beträgt die Vergleichsfrequenz:

$$f_n = f_{VCO}/n \qquad (2)$$

Zur Veränderung der Freilauffrequenz f_o ist eine Steuerspannung u_g erforderlich. u_g ist das Ausgangssignal eines Tiefpaßfilters.

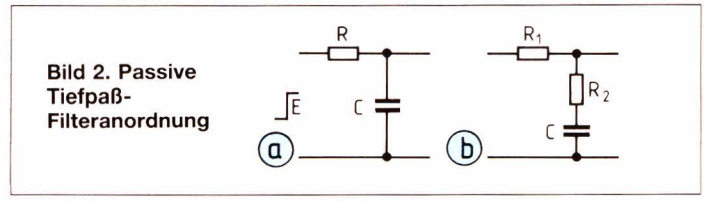


Bild 2. Passive Tiefpaß-Filteranordnung

Im Regelfall benutzt man Tiefpaßfilter aus passiven Bauelementen (*Bild 2*). Wenn an der Schaltung a ein Spannungssprung mit der Amplitude E anliegt, ergeben sich folgende Beziehungen:

$$E = i \cdot R + \frac{1}{C} \int i \, dt$$

$$0 = R \frac{di}{dt} + \frac{i}{C}$$

Die klassischen Lösungen dafür sind:

$$i = \frac{E-A}{R} \cdot e^{-\frac{t}{RC}} \qquad (3)$$

$$u_c = E-(E-A) \cdot e^{-\frac{t}{RC}} \qquad (4)$$

A ist dabei die Spannung über dem Kondensator, bevor die Sprungfunktion beginnt. Wenn man die Filteranordnung nach b verwendet, muß man lediglich R durch ($R_1 + R_2$) ersetzen.

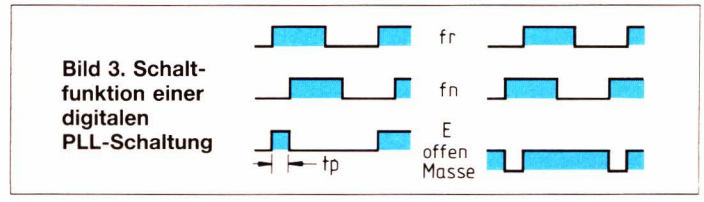


Bild 3. Schaltfunktion einer digitalen PLL-Schaltung

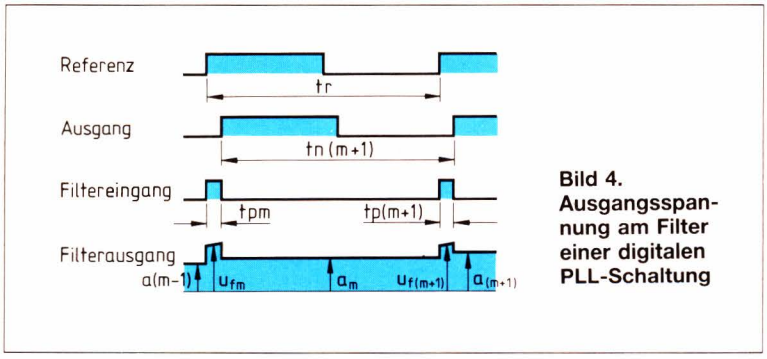


Bild 4. Ausgangsspannung am Filter einer digitalen PLL-Schaltung

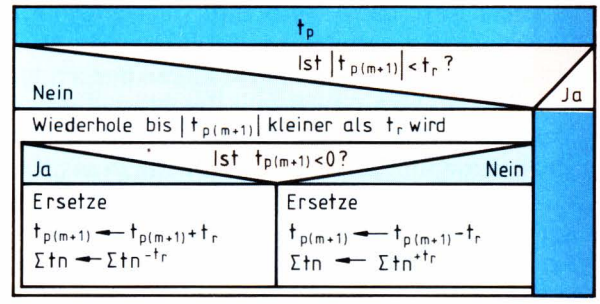


Bild 5. Korrektur des Wertes der Phasendifferenz

Ein- und Ausgang des Tiefpasses

Wenn ein digitaler Phasendetektor Verwendung findet, liegt am Eingang der Filteranordnung kein kontinuierliches Signal an. Während des größten Teils der Zeit ist der Filtereingang offen und an Ein- sowie Ausgang liegt die Spannung A an (die Restspannung über dem Kondensator).

Die steigenden Flanken von f_r und f_n steuern einen Schalter (*Bild 3*). Wenn die positive Flanke von f_r zeitlich vor f_n liegt, wird der Filtereingang offen geschaltet. Kommt die steigende Flanke von f_n vorher, wird der Filtereingang auf Masse bzw. Nullpotential geschaltet, f_r schaltet den Eingang wieder offen.

Für die Simulation wurde der Filtertyp B gewählt, weil diese Schaltung auch den Typ A beinhaltet, wenn man $R_2 = 0$ setzt. Man setzt voraus:

$$\tau_1 = R_1 \cdot C \text{ und } \tau_2 = R_2 \cdot C \tag{5}$$

Die Eingangsimpulse des Filters haben die Impulsdauer t_p. Während dieser Zeit nimmt die Spannung über dem Kondensator einen neuen Wert an. Nachdem der Eingangsimpuls beendet ist, beträgt die Spannung über dem Kondensator C:

$$u_c = E-(E-A) \cdot e^{-\frac{t_p}{\tau_1 + \tau_2}} \tag{6}$$

Während der Impuls anliegt, beträgt die Ausgangsspannung nicht nur u_c, sondern

$$u_f = E - i \cdot R_1 \tag{7}$$

$$u_f = E - (E-A) \frac{\tau_1}{\tau_1 + \tau_2} e^{-\frac{t_p}{\tau_1 + \tau_2}} \tag{8}$$

wobei $0 \leq t \leq t_p$ ist.

Am Ausgang des Filters ergibt sich somit ein Signal, wie es in *Bild 4* dargestellt ist. Weil in (6) U_c nur der neue Wert von a berechnet ist und die Zahl eines Impulses zum Schalten des Filtereinganges berücksichtigt werden muß, lassen sich nun die Gleichungen (6), (7) und (8) in die endgültige Form bringen:

$$a_m = E - (E-a_{(m-1)}) \cdot e^{-\frac{|t_{pm}|}{\tau_1 + \tau_2}} \tag{9}$$

$$u_{fn} = E - (E-a_{(m-1)}) \frac{\tau_1}{\tau_1 + \tau_2} e^{-\frac{t}{\tau_1 + \tau_2}} \tag{10}$$

Wobei $0 \leq t \leq |tpm|$ ist (11)

Ausgangsverhalten des Filters

Wie man aus Bild 4 erkennen kann, ändert sich die Ausgangsspannung am Filter sprungartig. Allerdings verändert ein VCO eine Frequenz nicht in unendlich kurzer Zeit. Daher benötigt man für die erste Simulation den Mittelwert der Filterausgangsspannung, um die neue Ausgangsfrequenz des VCO bestimmen zu können. Mit Hilfe der Beziehung (2) ergibt sich:

$$t_{n(n+1)} = \frac{n}{k_0 \cdot u_{gm} + f_0} \tag{12}$$

Für u_{gm} läßt sich annehmen:

$$u_{gm} = u_{fm} \cdot \frac{|t_{pm}|}{t_{nm}} + a_m \cdot \frac{t_{nm} - |t_{pm}|}{t_{nm}}$$

$$u_g = a_m + (u_{fm} - a_m) \frac{|t_{pm}|}{t_{nm}} \tag{13}$$

```
  5 GO SUB 500
 10 DATA 1,2e5,1.99e5,1e3,8e4,0.012,0.005,32e3
 20 READ st,f1,f2,fr,f0,t1,t2,kv
 25 LET n=f2/fr
 30 LET tr=1/fr: LET m=0: LET i=-1
 35 LET a=(f1-f0)/kv: LET tn=tr
 40 LET stn=1/(f1/n)
100 LET m=m+1
110 LET tp=m*tr-stn
120 IF tp=0 THEN  STOP
130 IF ABS tp<tr THEN  GO TO 180
140 IF tp<0 THEN  GO TO 160
150 LET tp=tp-tr: LET stn=stn+tr: GO TO 120
160 LET tp=tp+tr: LET stn=stn-tr: GO TO 120
180 LET e=0
190 IF tp<0 THEN  LET e=5
200 LET tpa=ABS tp
210 LET uc=e-(e-a)*(1-tpa/(t1+t2))
220 LET ug=uc+(e-a)*(t2*tpa/(t1+t2))/tn
225 LET tn=n/(kv*ug+f0): LET a=uc
230 LET stn=stn+tn: LET fn=1/tn
240 IF (m-INT (m/st)*st)<>0 THEN  GO TO 100
250 LET i=i+1
260 PLOT i,(fn*n-f1)/10+140
270 IF i<>250 THEN  GO TO 100
280 STOP
500 PLOT 0,0: DRAW 255,0: PLOT 0,0: DRAW 0,175
505 PLOT 0,0: DRAW 255,0: PLOT 0,0: DRAW 0,175
510 FOR i=0 TO 170 STEP 10
520 PLOT 0,i: DRAW 5,0: NEXT i
530 FOR i=0 TO 250 STEP 10
540 PLOT i,0: DRAW 0,5: NEXT i
550 RETURN
```

Bild 6. Listing des PLL-Simulationsprogrammes. Dieses Programm ist in BASIC für einen Rechner vom Typ Sinclair Spectrum geschrieben

Jetzt läßt sich das Formelsystem in die Programmsimulation umsetzen. e^x kann man durch die Taylor-Reihe darstellen:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots$$

Wenn x sehr klein im Vergleich zu 1 ist, läßt sich e^x durch 1 + x annähern. Es ist bekannt, daß $t_{pmax} = t_r$ und $t_p \ll (\tau_1 + \tau_2)$ sind. Daher läßt sich folgender Ersatz vornehmen:

$$e^{-\frac{|t_{pm}|}{\tau_1 + \tau_2}} \text{ durch } \left(1 - \frac{|t_{pm}|}{\tau_1 + \tau_2}\right)$$

Als endgültige Beziehung für die Simulation ergibt sich dann:

$$u_{gm} = E - (E - a_{(m+1)}) \left(1 - \frac{|t_{pm}|}{\tau_1 + \tau_2}\right) + (E - a_{(m+1)}) \left(\frac{\tau_2 \, |t_{pm}|}{\tau_1 + \tau_2}\right) \frac{1}{t_{nm}} \qquad (14)$$

Die Kombination von (14) und (12) ergibt den neuen Wert bei t_n, die zeitlich gesehen eine Referenzperiode später folgt.

Für alle Berechnungen ist die Impulslänge t_p erforderlich, die sich nach folgender Methode bestimmen läßt:

$$t_{p(m+1)} = (m+1) \cdot t_r - \sum_{i=1}^{m+1} t_{ni} \qquad (15)$$

Weil $0 \leq t_{p(m+1)} \leq t_r$ gilt, könnte es notwendig werden, das mit der Beziehung (15) bestimmte Resultat zu korrigieren. Dies erfolgt nach der in *Bild 5* gezeigten Methode.

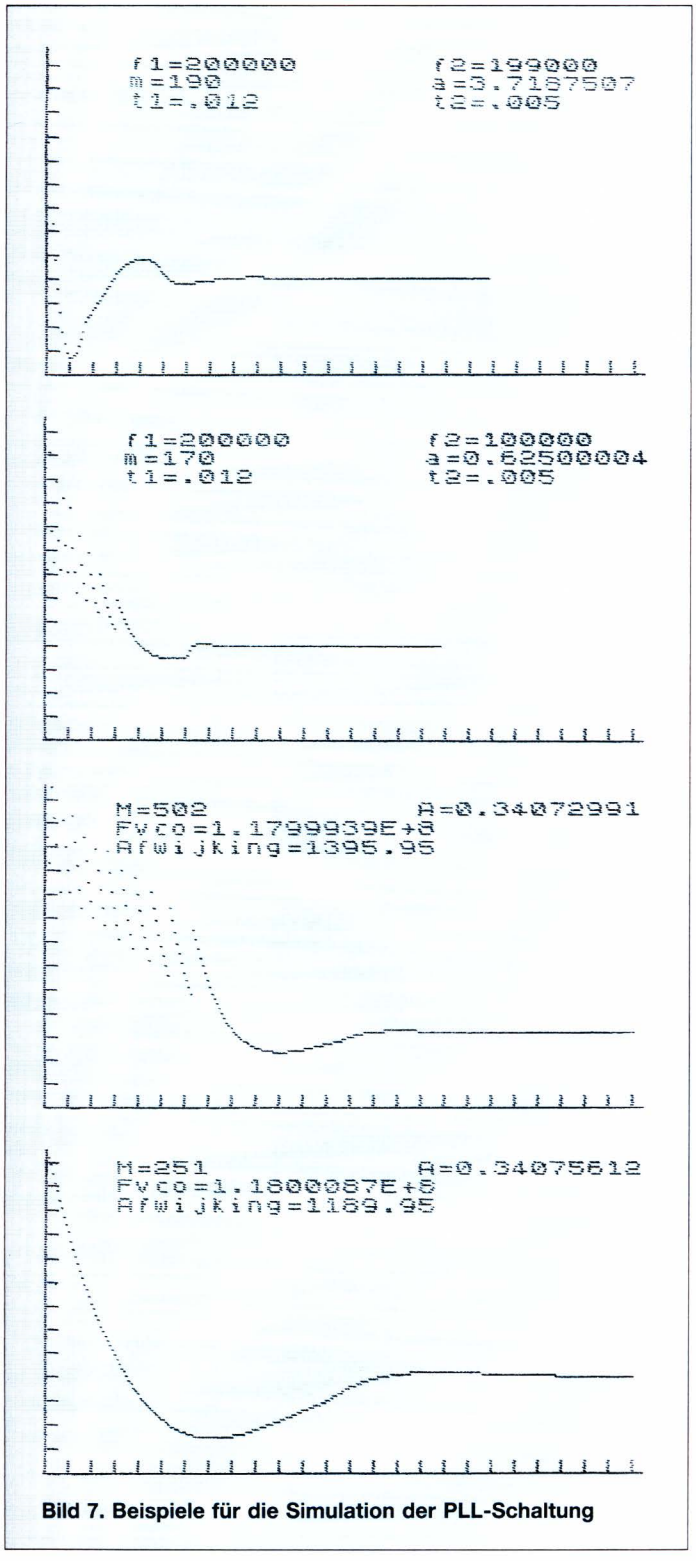


Bild 7. Beispiele für die Simulation der PLL-Schaltung

Simulationsprogramm

Das eigentliche Programm wurde in BASIC für einen Computer vom Typ Sinclair Spectrum geschrieben. *Bild 6* zeigt das Listing, das das Programm zur Bestimmung des Verhaltens einer PLL-Schaltung vom Typ 4046 errechnet. Der Synthesizer erzeugt eine Frequenz von 200 kHz in Schritten von 1 kHz.

Die Programmzeilen 100...230 ergeben sich aus den beschriebenen mathematischen Funktionen, während die Zeilen 240...280 der Ausgabe der gewünschten Kurve dienen. Falls erforderlich, kann die Zeile 260 so geändert werden, daß sie auch für Bildschirme mit 255 × 176 Pixeln geeignet ist.

Relevante Daten für die Berechnung werden mit Hilfe von Data-Statements in das Programm eingegeben.

st: die Anzahl der Schritte zwischen den dargestellten Punkten
f_1: f_{VCO} am Beginn der Berechnung
f_2: erforderliche neue VCO-Frequenz
f_r: Referenzfrequenz
f_0: Freilauffrequenz des VCO
t_1 und t_2: die Zeitkonstanten $t_1 = R_1 \cdot C$ und $t_2 = R_2 \cdot C$
k_v: VCO-Konstante (Hz/V)

Einzige Ausnahme ist die Spannung E, die in Zeile 190 einzugeben ist. *Bild 7* zeigt einige Resultate, die bei der Simulation von PLL-Schaltungen gewonnen wurden. Praktische Messungen haben gezeigt, daß die Signalverläufe mit praktisch gemessenen Ergebnissen sehr genau übereinstimmen.

Ing. Hendrik Mortier

Universeller Quadratwurzel-Algorithmus

Es ist ein Verfahren bekannt, das nach einem einfachen Subtraktions-Algorithmus die Berechnung der Quadratwurzel aus einem 16-Bit-Argument zuläßt. Diese Lösung macht Gebrauch von den 16-Bit-Befehlen des Prozessors Z80. Sie ist deshalb nicht unmittelbar auf andere 8-Bit-Mikroprozessoren übertragbar. Außerdem erfordert sie einen zusätzlichen Rundungs-Algorithmus und liefert kein 9-Bit-Resultat. Das nachstehend beschriebene Verfahren ist auf beliebige μP-Typen übertragbar. Es berechnet unmittelbar die gerundete Quadratwurzel, ist besonders schnell und berücksichtigt eine Laufzeitoptimierung durch Vorgabe mehrerer Startwerte.

Die Reihensumme

$$S_n = 1 + \sum_{k=1}^{n} 2(k-1) = 1 + 2 + 4 + 6 + 8 + \ldots + 2(n-1)$$

durchläuft die Folge der kleinsten ganzzahligen Argumente, deren Wurzel auf n aufzurunden ist, d. h. $n - 0{,}5 \leqslant \sqrt{S_n} \leqslant n$.

Die Allgemeingültigkeit dieser „Rundungsreihe" erkennt man leicht aufgrund folgender Überlegung: Nach dem Bildungsgesetz der Reihe muß $S_{n+1} = S_n + 2n$ gelten.

Wegen $(n - 0{,}5)^2 = n^2 - n + 0{,}25$ ist $S_n = n^2 - n + 1$ das kleinste Argument mit Wurzelwert n, wegen $(n + 0{,}5)^2 = n^2 + n + 0{,}25$ ist $S_{n+1} = n^2 + n + 1$ kleinstes Argument mit Wurzelwert n + 1, damit folgt $S_{n+1} = n^2 + n + 1 = n^2 - n + 1 + 2n = S_n + 2n$ entsprechend dem Bildungsgesetz der Reihe.

Subtrahiert man vom Argument, abgesehen von der ersten Eins, fortlaufend zunehmende gerade Zahlen, bis das Argument negativ wird, so erhält man aus der Anzahl der möglichen Subtraktionen unmittelbar den richtig gerundeten Quadratwurzelwert. Eine kleine Zusatzüberlegung führt nun zu dem besonders bequem durchführbaren Algorithmus. Die Reihe

$$\frac{S_n - 1}{2} = \sum_{k=1}^{n} (k-1) = 0 + 1 + 2 + 3 + 4 + \ldots + (n-1)$$

durchläuft die Folge der natürlichen Zahlen.

Teilt man zunächst das um 1 verminderte Argument durch 2 (1 Bit nach rechts schieben), so genügt es, fortlaufende natürliche Zahlen zu subtrahieren. Endwert des durch einfaches Inkrementieren zu bildenden Subtrahenden und gerundeter Quadratwurzelwert sind identisch.

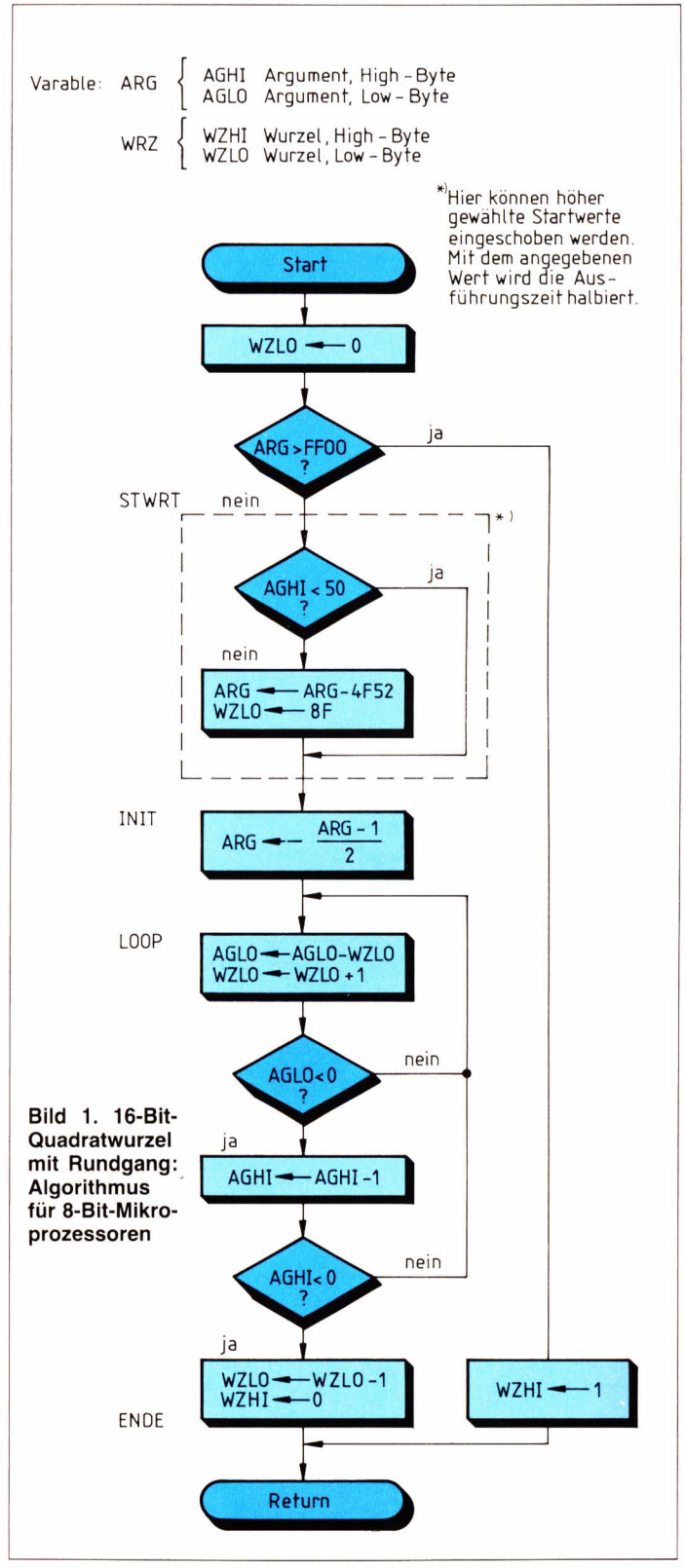


Bild 1. 16-Bit-Quadratwurzel mit Rundgang: Algorithmus für 8-Bit-Mikroprozessoren

```
* = $E0
00E0                TBL1    .BYT    $17,    $50,    $A1     ; Vergleichswerte
00E3                TBL2    .BYT    $4D,    $8F,    $CB     ; Wurzelwerte
00E6                TBL3    .BYT    $DC,    $52,    $2E     ; Subtrahend, Low-Bytes
00E9                TBL4    .BYT    $16,    $4F,    $A0     ; Subtrahend, High-Bytes
                    AGHI =  $F0             ; Argument, High-Byte
                    AGLO =  $F1             ; Argument, Low-Byte
                    WZHI =  $F2             ; Wurzel, High-Byte
                    WZLO =  $F3             ; Wurzel, Low-Byte
* = $200
0200    A200        START   LDX     00
0202    86F3                STX     WZLO    ; Wurzel, Low-Byte auf Null setzen
0204    A5F0                LDA     AGHI    ; Argument aus Speicher
0206    A4F1                LDY     AGLO    ; in Prozessor-Register laden
0208    F004                BEQ     STWRT   ; Wurzel < $100, zur Berechnung
020A    AA                  TAX
020B    E8                  INX
020C    F020                BEQ     ENDE    ; Wurzel = $100, zum Ende
020E    A202        STWRT   LDX     =$02    ; Tabellenindex laden
0210    D5E0        VGL     CMP     TBL1,X  ; Argument < Tabellenwert?
0212    B005                BCS     SUB     ; falls nein, subtrahieren
0214    CA                  DEX             ; falls ja, Index–1
0215    10F9                BPL     VGL     ; nächster Tabellenwert
0217    300C                BMI     INIT    ; oder Wert verarbeiten
0219    48          SUB     PHA
021A    B5E3                LDA     TBL2,X  ; Wurzel, Low-Byte
021C    85F3                STA     WZLO    ; aus Tabelle laden
021E    98                  TYA             ; und zugehörigen
021F    F5E6                SBC     TBL3,X  ; Subtrahenden
0221    A8                  TAY             ; aus Tabelle
0222    68                  PLA             ; vom Argument
0223    F5E9                SBC     TBL4,X  ; subtrahieren
0225    4A          INIT    LSR     .A
0226    AA                  TAX             ; Argument-High/2 nach Indexregister X
0227    98                  TYA             ; Argument-Low/2 nach Akkumulator
0228    6A                  ROR     .A      ; Carry bereitet Subtraktion von 1 vor
0229    E5F3        LOOP    SBC     WZLO    ; Argument – Wurzel
022B    E6F3                INC     WZLO    ; Wurzel + 1
022D    B0FA                BCS     LOOP    ; Wiederholung, falls kein Subtr.-Übertrag
022F    38                  SEC             ; Subtraktionsübertrag
0230    CA                  DEX             ; Argument-High – 1
0231    10F6                BPL     LOOP    ; Wiederholung, falls Argument-High ≥0
0233    C6F3                DEC     WZLO    ; Endwert Wurzel-Low herstellen
0235    E8          ENDE    INX             ; Endwert Wurzel-High herstellen (0 bzw. 1)
0236    86F2                STX     WZHI
0238    60                  RTS             ; Fertig
```

Bild 2. Berechnung der Quadratwurzel mit dem Prozessor 6502. Das Teilprogramm STWRT (bis INIT) kann entfallen. Die Ausführungszeit beträgt dann bei 1 MHz Taktfrequenz maximal 3,627 ms. Andernfalls reduziert sich die maximale Ausführungszeit auf rund 1 ms. Dies kommt durch die Startwerttabelle zustande, die beliebig erweitert werden kann. Das Prinzip wird hier anhand von drei Startwerten gezeigt:
Bei Argumenten ≥$A100 wird $A02E subtrahiert und WZL0 = $CB gesetzt,
bei Argumenten ≥$5000 wird $4F52 subtrahiert und WZL0 = $8F gesetzt,
bei Argumenten ≥$1700 wird $16DC subtrahiert und WZL0 = $4D gesetzt.
Grundsätzlich wird die größtmögliche Zahl subtrahiert, deren Wurzel gerade noch auf n–1 abgerundet werden müßte; die Wurzel wird auf n gesetzt

In *Bild 1* ist das Flußdiagramm für den Algorithmus angegeben. Es enthält keine prozessorspezifischen Besonderheiten. Zeitbestimmend ist die Schleife LOOP, sie muß deshalb so effektiv wie möglich codiert werden, d. h. soviele Variable wie möglich sollten in Prozessorregistern verfügbar sein, und die Zahl der Vergleiche muß so gering wie möglich gehalten werden. Aus dem zuletzt genannten Grund wird die Schleife mit einem Wurzelwert verlassen, der um eins zu groß ist, was außerhalb der Schleife leicht korrigiert werden kann.

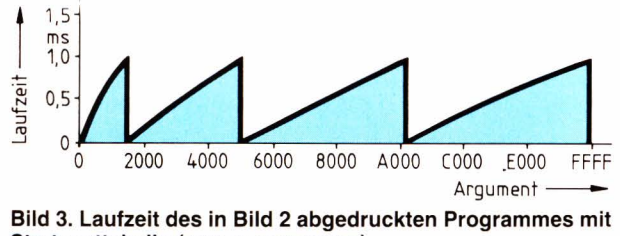


Bild 3. Laufzeit des in Bild 2 abgedruckten Programmes mit Startwerttabelle (genau gemessen)

Argumente größer als FF00 ergeben die gerundete Quadratwurzel 100 (hex.), dieser Fall wird gesondert und damit besonders schnell bearbeitet. Stehen 16-Bit-Befehle zur Verfügung, läßt man einfach die gesonderten Operationen und Vergleiche für das High-Byte von Argument und Wurzel entfallen. Die einschiebbare Teilroutine STWRT zeigt, wie man durch vorgegebene Startwerte zu einer verkürzten Ausführungszeit kommen kann. Wählt man n als Startwert für die Wurzel, so ist n^2-n vom Argument zu subtrahieren.

In *Bild 2* ist ein Programmbeispiel für den Prozessor 6502 angegeben. Zu bemerken ist dazu, daß durch die komplementäre Carry-Funktion bei Addition und Subtraktion die Berechnung von (ARG-1)/2 mit der Rechtsschiebeoperation beginnen kann. Eine anschließende Subtraktion berücksichtigt im Carry-Inhalt die eigentlich vorher notwendige Subtraktion von 1. Bei anderen Prozessoren ist die üblicherweise umgekehrte Carry-Funktion bei der Subtraktion zu beachten! Weitere Übertragungsprobleme dürften kaum auftreten. *Bild 3* zeigt die Ausführungszeiten.

Peter Rix

Schnelle Fouriertransformation in BASIC

Bei der Spektralanalyse der verschiedensten Vorgänge spielt die Fouriertransformation eine hervorragende Rolle im Ingenieurwesen und in der angewandten Mathematik. Im folgenden wird kurz auf ihre Grundlagen eingegangen. Danach wird die Wirkungsweise der schnellen Fouriertransformation (FFT, Fast Fourier Transform) skizziert und ein exemplarisches BASIC-Programm dazu vorgestellt. Die FFT wurde von J. W. Cooley und J. U. Tukey entwickelt und 1965 veröffentlicht [1].

Eine zeitlich periodische Funktion x(t) läßt sich unter wenigen, in der Praxis erfüllten Voraussetzungen in eine Fourierreihe zerlegen.

Die Periodizität wird ausgedrückt durch

$$x(t) = x(t + kT) \qquad (1)$$

mit k = beliebige ganze Zahl und T = Periode.

Die Formulierung der Funktion als Fourierreihe lautet

$$x(t) = \sum_{n=0}^{\infty} a_n \cos(2\pi nt/T) + b_n \sin(2\pi nt/T) \qquad (2)$$

oder in komplexer Schreibweise

$$x(t) = \sum_{n=0}^{\infty} X_n \exp(2\pi int/T) \qquad (3)$$

mit X_n = Fourierkoeffizienten, $i^2 = -1$, $Re(X_n) = a_n$, $Im(X_n) = b_n$ und $\exp(ix) = \cos(x) + i \cdot \sin(x)$. (Re, Im = Real- bzw. Imaginärteil)

Die auftretenden Frequenzen $2\pi n/T$ sind ganzzahlige Vielfache der Grundfrequenz $2\pi/T$.

Meistens liegt eine zeitabhängige Funktion jedoch nicht kontinuierlich, sondern als leicht von einem Rechner zu verarbeitende Zeitreihe $x_0, x_1, x_2, \ldots, x_{n-1}$ vor. N ist die Anzahl der Meßpunkte. Die Zeitreihe ist das Ergebnis der Digitalisierung eines analogen Signals in bestimmten Zeitabständen Δt. $N\Delta t$ entspricht dann der Grundperiode T. Das k-te Glied der Reihe ist $x_k = x(t_0 + k\Delta t)$ mit k = 0, 1, 2... und t_0 = Beginn der Messung. Die Periodizität (vgl. (1)) wird ausgedrückt durch:

$$x_k = x_{k+jN} \qquad j = 0, \pm 1, \pm 2 \ldots$$

Die Fourierkoeffizienten der DFT (*Discrete Fourier Transform*) lauten dann

$$X_n = (1/N) \sum_{j=0}^{N-1} x_j W^{nj} \qquad (4)$$

mit n = 0, 1, 2...N–1 und $W = \exp(-2\pi i/N)$.

Die Inverse von (4) ist (5)

$$x_n = \sum_{j=0}^{N-1} X_j W^{-nj}$$

(Die Definition der DFT nach Gl. (4) und (5) ist nicht einheitlich in der Literatur. Zum Beispiel wird manchmal der Normierungsfaktor (1/N) in (5) statt in (4) verwendet, oder der Exponent in W bekommt ein anderes Vorzeichen)

Dabei stellt

$$Re(X_n) = (1/N) \sum_{j=0}^{N-1} x_j \cos(2\pi jn/N)$$

den symmetrischen (geraden) und

$$Im(X_n) = (1/N) \sum_{j=0}^{N-1} x_j \sin(2\pi jn/N)$$

den antisymmetrischen (ungeraden) Teil des Spektrums dar. Da nach Gl. (4) für jeden der N Fourierkoeffizienten X_n eine Summe aus N Summanden erforderlich ist, ist die Gesamtzahl der erforderlichen Multiplikationen $\sim N^2$, d. h. bei N = 1024 etwa eine Million.

Die schnelle Fouriertransformation (FFT) schafft hier Abhilfe. Ihre Wirkungsweise besteht in einer wiederholten Aufspaltung der Zeitfunktion in gerade und ungerade Terme. Im folgenden sei vorausgesetzt, daß die Anzahl der Daten N eine Potenz von 2 sei, z. B. $N = 2^{10} = 1024$.

Die erste Aufspaltung geht so vor sich: Zunächst wird (4) umgeformt:

$$X_n = (1/N) \sum_{j=0}^{N/2-1} (x_{2j} W^{2jn} + x_{2j+1} W^{(2j+1)n})$$

oder

$$X_n = (1/N) \sum_{j=0}^{N/2-1} x_{2j} W^{2jn} + W^n (1/N) \sum_{j=0}^{N/2-1} x_{2j+1} W^{2jn} \qquad (6).$$

Zahl der Multiplikationen bei DFT und FFT

K	$N=2^K$	DFT N^2	FFT 2*K*E
2	4	16	16
3	8	64	48
4	16	256	128
5	32	1024	320
6	64	4096	768
7	128	16384	1792
8	256	65536	4096
9	512	262144	9216
10	1024	1048576	20480
11	2048	4194304	45056
12	4096	16777216	98304
13	8192	67108864	212992
14	16384	268435456	458752

X_n kann also durch die Fouriertransformierten Y_n und Z_n der Zeitreihen y_n und z_n dargestellt werden, die aus x_n gemäß $y_n = x_{2n}$ und $z_n = x_{2n+1}$ gewonnen werden:

$$X_n = Y_n + W^n Z_n \qquad (7)$$

mit

$$Y_n = (1/N) \sum_{j=0}^{N/2-1} y_n W^{2jn} \text{ und}$$

$$Z_n = (1/N) \sum_{j=0}^{N/2-1} z_n W^{2jn}$$

Da die Reihen y_n und z_n aus je N/2 Zahlenwerten bestehen, ist der Rechenaufwand ca. $(N/2)^2 + (N/2)^2 = N^2/2$ und hat sich somit gegenüber einer DFT nach Gl. (4) halbiert! Die FFT setzt dieses Verfahren fort, indem y_n und z_n selbst wieder aufgespalten werden usw. Der insgesamt für die FFT aufzubringende Rechenaufwand reduziert sich damit weiter bis auf $\sim 2N \cdot \mathrm{ld}\,N = 2\,N\,k$ bei $N = 2^k$ Elementen der Zeitreihe x (ld = Logarithmus zur Basis 2). Einen Eindruck von der Zeitersparnis beim Rechnen mit der FFT vermittelt die Tabelle mit der ungefähren Anzahl der nötigen Multiplikationen. Die tatsächliche Zeitersparnis ist kleiner als die rechnerisch aus der *Tabelle* ermittelte, weil die FFT gegenüber der DFT einige Hilfsoperationen erfordert.

Das abgedruckte Programm liest eine Reihe von Daten aus den DATA-Anweisungen, führt eine FFT durch (Schalter LS = 1) und druckt das Ergebnis aus. Dann wird eine Rücktransformation (Schalter LS = –1) durchgeführt und ebenfalls das Ergebnis ausgedruckt. Die Rück-FFT rekonstruiert aus den Fourierkoeffizienten (Frequenzbereich) wieder die ursprüngliche Zeitreihe. Das Programm ist für komplexe Zeitreihen ausgelegt. Bei der Berechnung reeller Zeitreihen braucht nur wie im Beispiel der Imaginärteil gleich Null gesetzt zu werden.

Dr. Ulrich Breymann

Literatur

[1] *Cooley, J. W., Tukey, J. U.:* An algorithm for the machine calculation of complex Fourier series. Math. of Comput. 19, April 1965, S. 297...301.

[2] *Cochran, W. T. et al:* What is the Fast Fourier Transform? IEEE Transactions on Audio and Electroacoustics, Bd. AU–15, Nr. 2, S. 45...55, Juni 1967.


```
FOURIERTRANSFORMATION

NR.              REAL-
 1                 4.5
 2                 -.5
 3                 -.5
 4                 -.5
 5                 -.5
 6                 -.5
 7                 -.5
 8                 -.5

RUECKTRANSFORMATION

NR.              REAL-
 1                   1
 2                   2
 3                   3
 4                   4
 5                   5
 6                   6
 7                   7
 8                   8


IMAGINAERTEIL
  0
 -1.2071
 -.5
 -.2071
  0
  .2071
  .5
  1.2071


IMAGINAERTEIL
  0
  0
  0
  0
  0
  0
  0
  0
```

```
10 REM ********  SCHNELLE FOURIER-TRANSFORMATION **********
20 REM
30 DEFINT I-N :REM *  MIT I BIS N BEGINNENDE VARIABLEN SIND GANZZAHLIG
40 DIM X(2,1024),M(13)
50 INPUT"LD N   (Z.B. 10 FUER 2↑10 = 1024 DATENPUNKTE)";N
60 REM *  EINLESEN DER DATEN:   1. REALTEIL
70 FOR I = 1 TO 2↑N : READ X(1,I) : NEXT I
80 REM                            2. IMAGINAERTEIL
90 FOR I = 1 TO 2↑N : READ X(2,I) : NEXT I
100 REM
110 LPRINT"FOURIERTRANSFORMATION":LPRINT" "
120 LS=1
130 GOSUB 200 : REM *  FFT
140 GOSUB 390 : REM *  ERGEBNIS AUSDRUCKEN
150 LPRINT" ":LPRINT"RUECKTRANSFORMATION":LPRINT" "
160 LS=-1
170 GOSUB 200 : REM *  FFT
180 GOSUB 390 : REM *  ERGEBNIS AUSDRUCKEN
190 END
200 REM **** SUBROUTINE FAST FOURIER TRANSFORM (FFT) ****
210 P=2*3.14159:M(N)=1:FORI=2TON:M(N-I+1)=M(N-I+2)*2:NEXTI
220 FORI0=1TON:I1=0:M0=M(I0)
230 FORI2=1TOM(N-I0+1):F1=I1:F2=2*M(I):F3=LS*P*F1/F2:F4=COS(F3)
240 F5=SIN(F3):I3=2*M0*(I2-1)
250 FORI=1TOM0:I4=I3+I:I5=I4+M0:F3=X(1,I5)*F4-X(2,I5)*F5
260 F1=X(2,I5)*F4+X(1,I5)*F5:X(1,I5)=X(1,I4)-F3:X(2,I5)=X(2,I4)-F1
270 X(1,I4)=X(1,I4)+F3:X(2,I4)=X(2,I4)+F1:NEXTI
280 FORI=2TON:I5=I:IFI1(M(I)THEN300
290 I1=I1-M(I):NEXTI
300 I1=I1+M(I5):NEXTI2:NEXTI0
310 I1=0:FORI4=1TO2*M(1):IFI1(I4THEN340
320 F3=X(1,I4):F1=X(2,I4):X(1,I4)=X(1,I1+1):X(2,I4)=X(2,I1+1)
330 X(1,I1+1)=F3:X(2,I1+1)=F1
340 FORI=1TON:I5=I:IFI1(M(I)THEN360
350 I1=I1-M(I):NEXTI
360 I1=I1+M(I5):NEXTI4:IFLS(0THENRETURN
370 FORI=1TO2*M(1):X(1,I)=X(1,I)/F2:X(2,I)=X(2,I)/F2:NEXTI:RETURN
380 REM
390 REM **** SUBROUTINE AUSDRUCK ****
400 REM        (MIT RUNDUNG WEGEN DER NUR 6-STELLIGEN GENAUIGKEIT
410 REM         DER WINKELFUNKTIONEN)
420 LPRINT"NR.              REAL-              IMAGINAERTEIL"
430 REM
440 FORI=1TO2↑N:LPRINTI,INT(10000*X(1,I)+.5)/10000,INT(10000*X(2,I)+.5)/10000:NE
XTI:RETURN
450 DATA 1,2,3,4,5,6,7,8
460 DATA 0,0,0,0,0,0,0,0
```

Das Programm liest die Kurvenwerte aus den DATA-Zeilen (450 und 460) und führt die FFT durch; mit einer Rücktransformation kann die Zeitreihe wieder rekonstruiert werden. Es läuft auf einem TRS-80 (zur Verfügung gestellt von der Computer Studio Braunschweig GmbH), kann aber leicht an andere Maschinen angepaßt werden

Mikrocomputer:

Beschleunigung der Multiplikation ohne zusätzliche Hardware

Multiplikation und Division sind im Befehlssatz der 8-Bit-Mikroprozessoren nicht vorhanden und können deshalb nur mit Hilfe von Makrobefehlen oder Unterprogrammen ausgeführt werden. In vielen Anwendungsfällen ist es erwünscht, die von den Mikroprozessor-Herstellern vorgeschlagenen Unterprogramme zu beschleunigen, ohne in die Hardware des Mikrorechners (z. B. durch Einbau eines Multiplizierers) einzugreifen. Der Autor stellt ein einfaches Verfahren vor, das die Multiplikation zweier 8-Bit-Operanden gegenüber den bekannten Verfahren drei- bis fünfmal schneller macht. Diese Methode wurde zuerst für den Mikroprozessor 8085 von Intel realisiert, und dann übertrug man die Programme auf den Z80 von Zilog. Im Prinzip kann dieses Verfahren auch für 16-Bit-Prozessoren verwendet werden.

Das Verfahren zur Beschleunigung der Multiplikation basiert auf folgender algebraischer Gesetzmäßigkeit:

$$a \times b = \frac{1}{4}\left[(a + b)^2 - (a - b)^2\right]$$

Dabei wird die Multiplikation auf eine Addition, zwei Subtraktionen und zwei Zugriffe zu einer vorher erstellten Quadrattafel zurückgeführt. Die Division durch 4 kann man in die Quadrattabelle einarbeiten ($x^2/4$). Um die Wortbreite der Tafel nicht auf 18 Bit erweitern zu müssen, muß eine Rundung durchgeführt werden, die aber zu keinem Rundungsfehler führt (siehe Kasten „Erzeugen der Quadrattafeln").

In *Bild 1* ist das Blockschaltbild der Multiplikation zweier binärer 8-Bit-Operanden mittels einer Quadrattafel ($x^2/4$) dargestellt, und *Bild 2* zeigt das zugehörige Flußdiagramm sowie das Programmlisting.

Adreßberechnung

Die Adreßberechnung dient der Vorbereitung des Speicherzugriffs auf die Quadrattabelle mittels des Doppelregisters HL, wobei Register H die Teiladresse QT_0 oder QT_1 (in Abhängigkeit vom Wert des Übertragsbits cy=0 bzw. 1) und Register L die Teiladresse a+b oder /a–b/ enthält. Durch QT_0 wird das Unterbyte und durch QT_0+1 das Oberbyte des 16 Bit breiten Quadrates für den ersten Bereich des Übertragsbits (cy=0) und durch QT_1 das Unterbyte und durch QT_1+1 das Oberbyte für den 2. Bereich des Übertragsbits (cy=1) adressiert. Legt man die Quadrattabelle in vier aufeinanderliegenden Speicherseiten an, so kann man durch zweimaliges Inkrementieren oder Dekrementieren des Registers H zwischen den Bereichen QT_0 und QT_1 umschalten.

Weitere Beschleunigung

Um den Ablauf der Multiplikation noch schneller zu machen, gibt es eine Reihe von Möglichkeiten. Anstelle

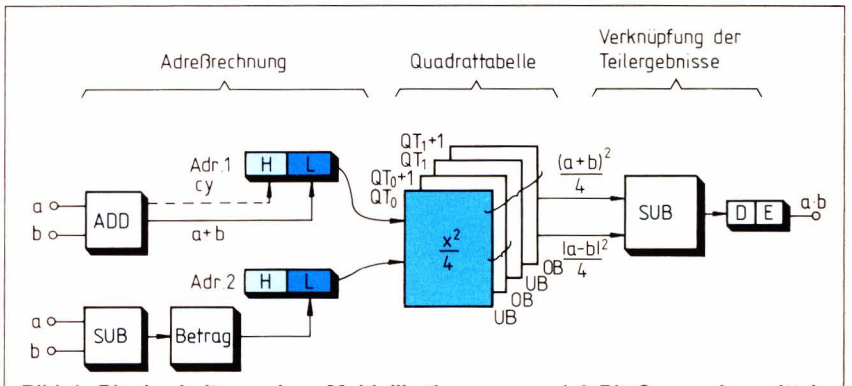


Bild 1. Blockschaltung einer Multiplikation von zwei 8-Bit-Operanden mittels einer Quadrattabelle $x^2/4$, das Flußdiagramm und Programmlisting zeigt Bild 2

der Berechnung des Betrags /a–b/ wird das Übertragsbit abgefragt und in Abhängigkeit seines Wertes das Programm verzweigt. Dies erfordert jedoch das Anlegen eines weiteren Bereichs der Quadrattafel (QT_2, QT_2+1), in dem die entsprechende Betragsbildung eingearbeitet ist. Wird bei der Subtraktion /a–b/ das Übertragsbit cy=1, so bedeutet das, daß das Ergebnis negativ ist, daß also 256 – /a–b/ zu bilden ist. Die Quadrattabelle ist also im Bereich QT_2, QT_2+1 anzulegen, so daß bei der Adresse 0 das Quadrat $0^2/4=0$, bei der Adresse 1 das Quadrat $255^2/4=16256$, bei der Adresse 2 das Quadrat $254^2/4=16129$ usw., bei der Adresse 254 das Quadrat $2^2/4=1$ und bei der Adresse 255 das Quadrat $1^2/4=0$ zu speichern ist.

Das Programm wird nach den Verzweigungen insgesamt zweimal im Programmspeicher abgelegt, um die Zeit für die Ausführung weiterer Rücksprünge zu vermeiden. Dadurch erreicht man einen symmetrischen Aufbau und eine von den Eingangsdaten unabhängige Laufzeit des Programms. Außerdem kann man durch geschickte Registerbelegung überflüssige Transportbefehle vermeiden (*Bild 4*). Je nach Befehlssatz und Prozessorstruktur können u. U. Befehle mit doppelter Wortbreite bei der Subtraktion der Teilergebnisse zu kürzeren Laufzeiten führen (Doppel-Addier- oder Doppel-Subtrahier-Befehle wurden in den abgebildeten Beispielen nicht verwendet).

Bei dem in *Bild 3* gezeigten Beispiel wurde die Adresse 1 im Doppelregister DE und Adresse 2 im Doppelregister HL abgelegt. Ist nach der Summenbildung (a+b) das Übertragsbit cy=0, so wird $D=QT_0$, anderen-

```
T0:     LD      BC,0        ;Normieren des Mikrorechners
        LD      DE,0
        LD      HL,1000H    ;QT0 = 10H
;
T1:     EX      DE,HL
        ADD     HL,BC       ;addiere laufenden Index i
        EX      DE,HL
        LD      (HL),DE
        INC     H
        LD      (HL),D      ;Quadrat in Tabelle          1. Teil
        INC     L                                        H = QT0 und QT0+1
;
        EX      DE,HL
        ADD     HL,BC       ;addiere laufenden Index i
        EX      DE,HL
        LD      (HL),D
        DEC     H
        LD      (HL),E      ;Quadrat in Tabelle
        INC     L
        INC     C
        JP      P,T1        ;Vorzeichenabfrage
;-----------------------------------------------------------------
        LD      HL,1200H    ;QT0 = 12H
;
T2:     EX      DE,HL
        ADD     HL,BC
        EX      DE,HL
        LD      (HL),E
        INC     H
        LD      (HL),D
        INC     L
;                           ;wie oben                    2. Teil
        EX      DE,HL                                    H = QT1 und QT1+1
        ADD     HL,BC
        EX      DE,HL
        LD      (HL),D
        DEC     H
        LD      (HL),E
        INC     L
        INC     C
        JP      NZ,T2       ;Nullabfrage

T3:     LD      DE,10FFH
        LD      HL,1401H
;
T4:     LD      A,(DE)                                   ;3. Teil durch Umspeichern
        DEC     E                                         aus 1. Teil,
        LD      (HL),A                                   ;zuerst Bereich
        INC     L                                            QT2
        JP      NZ,T4
;
        LD      DE,11FFH
        HL,1501H
;
T5:     LD      A,(DE)
        DEC     E                                        ;dann Bereich
        LD      (HL),A                                       QT2+ 1
        INC     L
        JP      NZ,T5
;
        XOR     A
        LD      (1400H),A                                ;Korrektur bei Adresse 0
        LD      (1500H),A
;
        HALT
;
```

Erzeugen der Quadrattabelle $x^2/4$. Dabei wurde BC als Indexregister für i, DE als Akkumulator für das Quadrat und HL als Speicherzeiger für die Quadrattabelle (abwechselnd mit DE) verwendet

Erzeugen der Quadrattafel

Im folgenden wird gezeigt, daß es möglich ist, die bei der Division durch 4 nach rechts geschobenen beiden niederwertigen Bits abzuschneiden und trotz des Rundungsfehlers die Multiplikation fehlerfrei auszuführen. Durch diese Rundung vermeidet man 18 Bit breite Quadrate. Die Quadrate gerader Integergrößen sind durch 4 teilbar. Bei Division der Quadrate durch 4 tritt also kein Rundungsfehler auf, wenn man die beiden niederwertigsten Bits (00) abschneidet. Bei ungeraden Integergrößen tritt jedoch ein Rundungsfehler auf. Es läßt sich jedoch zeigen, daß diese Bits immer eine 0 und 1 haben. Daher stört dieser Rundungsfehler nicht, weil er sich bei der Subtraktion der Teilergebnisse $(a+b)^2$ – $/a-b^2/$ heraushebt.

Sind sowohl a als auch b gerade bzw. ungerade, so sind (a+b) und (a–b) gerade. Wenn a gerade und b ungerade ist (oder umgekehrt), ist das Ergebnis der Addition bzw. Subtraktion immer ungerade. Ein möglicher Rundungsfehler hebt sich bei der Subtraktion der Teilergebnisse jedoch heraus, weil er bei beiden den gleichen Wert hat.

Tritt bei der Summe (a+b) ein Übertrag auf, so wird eine neue Speicherseite in einem erweiterten Teil der Quadrattafel adressiert. Da es sehr mühsam und fehleranfällig sein könnte, die Quadrattabellen von Hand anzulegen, wurde ein Programm zur automatischen Erzeugung der Tafel geschrieben. Für die Erzeugung der durch vier dividierten und gerundeten Quadrate wurde das abgebildete Programm verwendet. Es nutzt die Gesetzmäßigkeit aus, daß sich die Quadrate als Summe darstellen lassen:

$$n^2 = \text{ent}\left[\frac{n^2}{4}\right] = 0 + 1 + 1 + 2 + 2 + 3 + 3 + \ldots$$

Bei dem abgebildeten Programm enthält das Doppelregister BC den laufenden Indexwert i, DE dient als Akkumulator für die Quadrate und wird abwechselnd mit dem Speicherzeiger-Doppelregister HL benutzt. Zuerst wird der Tabellenbereich QT_0, QT_0+1, dann QT_1, QT_1+1 entsprechend der Darstellung in Bild 1 angelegt. Durch Umordnen der Quadrate im Bereich QT_1, QT_1+1 kann schließlich die Quadrattabelle QT_2, QT_2+1 erzeugt werden.

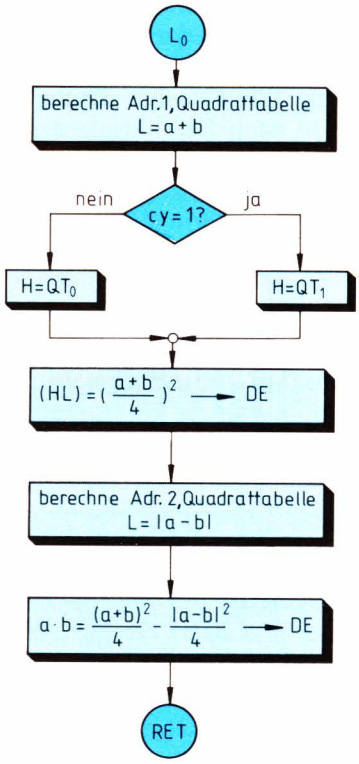


Bild 2. Flußdiagramm und Programmlisting. Dabei sind B der erste und C der zweite Operand, in BE werden Zwischen- und Endergebnisse abgespeichert, und HL ist der Speicherzeiger zur Quadrattabelle

				Programm-speicherzellen	States
L0:	LD	H, 11H	;H = QT_0 + 1	2	7
	LD	A,B,		1	4
	SUB	C	;(a - b)	1	4
	JP	NC, L1		3	10
	CPL			1	4
	INC	A	;Betrag	1	4
L1:	LD	L, A	;/a - b/ nach L	1	4
	LD	D, (HL)		1	7
	DEC	H		1	4
	LD	E, (HL)	;/a - b/²/₄ nach DE	1	7
;					
	LD	A, B		1	4
	ADD	A, C	;(a + b)	1	4
	LD	L, A		1	4
	JP	NC, L2		3	10
	INC	H		1	4
	INC	H	;wenn cy = 1: H = QT_1	1	4
L2:	LD	A, (HL)		1	7
	SUB	E		1	4
	LD	E,A		1	4
	INC	H	;$\frac{1}{4}$ [(a - b)² - /a - b/²] nach DE	1	4
	LD	A, (HL)		1	7
	SBC	A,D		1	4
	LD	D,A		1	4
	RET		;Rücksprung vom Unterpr.	1	10
			Summe	29	129

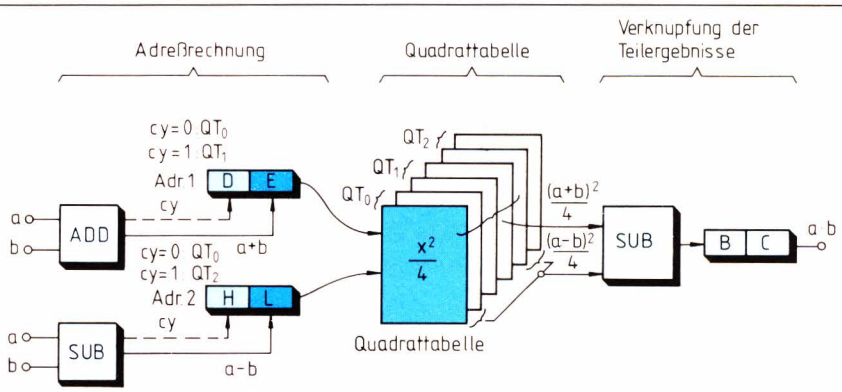
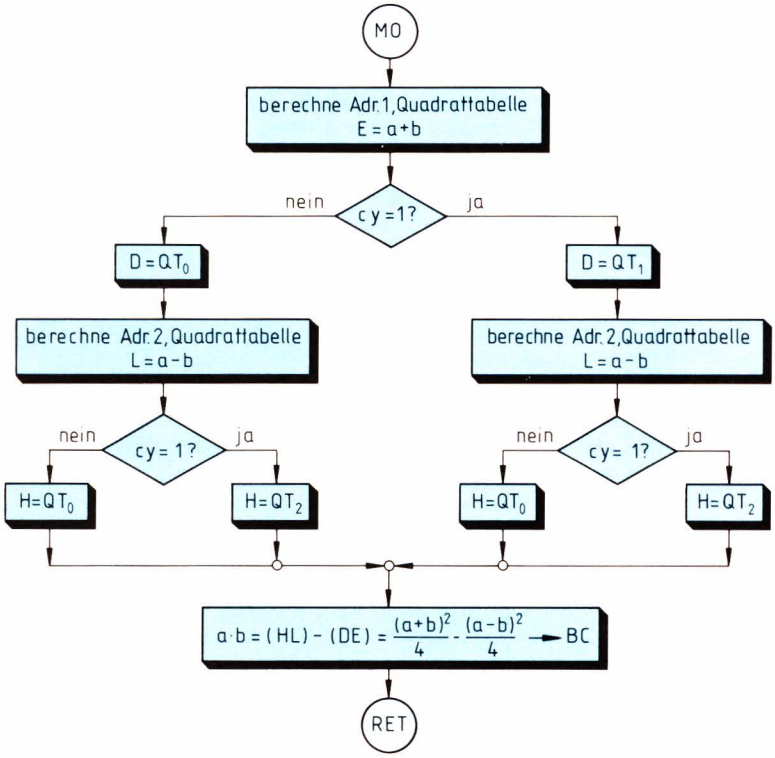


Bild 3. Blockschaltung einer beschleunigten Multiplikation, das zugehörige Flußdiagramm und Programmlisting zeigt Bild 4



				Programm-speicherzellen	States
L0:	LD	A,B		1	4
	ADD	A,C	;(a + b)	1	4
	LD	E,A		1	4
	JP	C,L1		3	10
;					(22)
	LD	D,10H	;cy = 0, D = QT_0	2	7
	LD	A,B		1	4
	SUB	C	;(a - b)	1	4
	LD	L,A		1	4
	JP	C,L2		3	10
;					(29)
L3:	LD	H,10H	;cy = 0, H = QT_0	2	7
	LD	A,(DE)		1	7
	SUB	(HL)		1	7
	LD	C,A		1	4
	INC	D		1	4
	INC	H		1	4
	LD	A,(DE)		1	7
	SBC	A,(HL)	;$\frac{(a+b)^2}{4} - \frac{(a-b)^2}{4}$ →BC	1	7
	LD	B,A		1	4
	RET		;Rücksprung vom Unterpr.	1	10
;					(61)
L1:	LD	D,12H	;cy = 1, D = QT_1	2	7
	LD	A,B		1	4
	SUB	C	;/a - b/	1	4
	LD	L,A		1	4
	JP	NC,L3		3	10
;					(29)
L2:	LD	H,14H	;cy = 1, H = QT_2	2	7
	LD	A,(DE)		1	7
	SUB	(HL)		1	7
	LD	C,A		1	4
	INC	D		1	4
	INC	H		1	4
	LD	A,(DE)		1	7
	SBC	A,(HL)	;$\frac{(a+b)^2}{4} - \frac{(a-b)^2}{4}$ →BC	1	7
	LD	B,A		1	4
	RET		;Rücksprung vom Unterpr.	1	10
;					(61)
			Summe	44	112

+ Tabelle 6 . 2^8 Zellen = 1,5 kByte

Bild 4. Flußdiagramm und Programmlisting der beschleunigten Multiplikation. In BC steht am Ende das Ergebnis, und in DE wird die Adresse 1 des Speicherzeigers und in HL die Adresse 2 abgelegt

falls $D=QT_1$ gesetzt. Ist nach der Differenzbildung a–b das Übertragsbit cy=0, so wird $H=QT_0$ und im umgekehrten Fall $H=QT_2$ gesetzt. Das Ergebnis wird im Doppelregister BC abgelegt. Das Programm und das zugehörige Flußdiagramm zeigt *Bild 4*.

Zum Schluß soll noch darauf hingewiesen werden, daß das Verfahren zur Multiplikation mittels Quadrattabelle durch mehrmalige serielle Anwendung auch auf Operanden mit größerer Wortbreite (z. B. 8 Bit * 16 Bit oder 16 Bit * 16 Bit) angewendet wurde. Dabei wurde von folgenden Beziehungen Gebrauch gemacht (Wortbreite von a und b 16 Bit):

$$a = 256\, a_1 + a_2$$
$$b = 256\, b_1 + b_2$$
$$ab = 256^2 a_1 b_1 + 256\, a_1 b_2 + 256\, a_2 b_1 + a_2 b_2.$$

Durch viermalige serielle Anwendung der Multiplikation 8 Bit * 8 Bit auf die Terme $a_1 b_1$, $a_1 b_2$, $a_2 b_1$ und $a_2 b_2$ und durch Berücksichtigung der Wertigkeiten (Faktor 256 bedeutet Verschiebung um 8 Bit, Faktor 256^2 bedeutet Verschiebung um 16 Bit) und entsprechende Addition der Teilergebnisse kann man auch hierbei einen ähnlich hohen Zeitgewinn erzielen wie bei den oben betrachteten Beispielen zu Multiplikation.

Dr.-Ing. Rainer Portscht

8080 – 8085 – Z80: Nicht ganz kompatibel

Für den Mikroprozessor 8080 existiert ein außergewöhnlich großes Softwareangebot (z. B. für das Betriebssystem CP/M). Da der Prozessor selbst jedoch veraltet ist, werden die Programme meistens auf den Nachfolgern 8085 und Z80 eingesetzt, wobei dann angenommen wird, daß sie zum 8080 softwaremäßig kompatibel sind, daß also alle Programme, die mit einem 8080 entwickelt wurden, in der gleichen Weise auch auf dem 8085 bzw. Z80 ablaufen. Daß dies nicht immer der Fall ist, zeigt dieser Beitrag.

Die Unterschiede betreffen die Manipulation des Parity-Flags und des Hilfs-Carry-Flags (für den DAA-Befehl). Beim Z80 wird das Parity-Flag bei einigen Befehlen als Overflow-Flag verwendet, während es beim 8080 immer die Parität des Wertes anzeigt. Da Parität und Überlauf etwas völlig verschiedenes darstellen, treten hier Inkompatibilitäten auf. *Tabelle 1* zeigt alle 8080-Befehle, die dieses Flag verändern. Bei den arithmetischen Operationen zeigt hier der Z80 ein anderes Verhalten. Um Kompatibilität zu erreichen, sollte daher bei 8080-Programmen das Parity-Flag nur nach den Operationen AND, OR und XOR abgefragt werden.

Der zweite Unterschied betrifft das Hilfs-Carry-Flag für den DAA-Befehl. Dieses Flag wird beim 8080 bei arithmetischen Operationen manipuliert und stellt den Übertrag von Bit 3 zu Bit 4 dar. Bei den anderen Operationen (außer bei DAA selbst) bleibt es dagegen unverändert. *Tabelle 2* zeigt, daß dies sowohl für den Z80 als auch den 8085 nicht gilt, obwohl gerade die Kompatibilität des letzteren zum 8080 hervorgehoben wird. Diese Einschränkung geschah vermutlich deshalb, um die Logik für die Befehlsdecodierung gegenüber dem 8080 zu minimieren. Da jedoch der DAA-Befehl nicht nur für

Tabelle 1. Unterschiedliche Behandlung des P-Flags (P/O-Flag beim Z80)

Assemblerbefehl 8080-Mnem.	Assemblerbefehl Z80-Mnem.	Verwendung des Parity-Flags 8080 bzw. 8085	Verwendung des Parity-Flags Z80
ADD s	ADD A,s	Parität	Überlauf
ADC s	ADC A,s	Parität	Überlauf
SUB s	SUB s	Parität	Überlauf
SBB s	SBC A,s	Parität	Überlauf
ANA s	AND s	Parität	Parität
ORA s	OR s	Parität	Parität
XRA s	XOR s	Parität	Parität
CMP s	CP s	Parität	Überlauf
DAA	DAA	Parität	Parität
INR r	INC r	Parität	Überlauf
DCR r	DEC r	Parität	Überlauf

s 8-Bit-Register oder 8-Bit-Daten
r 8-Bit-Daten

Tabelle 2. Abweichungen bei Veränderungen des AC- bzw. H-Flags

Assemblerbefehl 8080-Mnem.	Assemblerbefehl Z80-Mnem.	AC- bzw. H-Flag-Veränderung 8080	AC- bzw. H-Flag-Veränderung 8085	AC- bzw. H-Flag-Veränderung Z80
AND s	AND s	*	1	1
ORA s	OR s	*	0	1
XRA s	XOR s	*	0	1
DAD q	ADD HL,ss	*	*	?
RLC	RLCA	*	*	0
RRC	RRCA	*	*	0
RAL	RLA	*	*	0
RAR	RRA	*	*	0
CMA	CPL	*	*	1
STC	SCF	*	*	0
CMC	CCF	*	*	?
DAA	DAA	?	?	?

r 8-Bit-Daten
q 16-Bit-Register B, D, H, SP (8080-Mnem.)
ss 16-Bit-Register BC, DE, HL, SP (Z80-Mnem.).
* Flag wird nicht verändert
0 Flag wird zurückgesetzt
1 Flag wird gesetzt
? Flag wird verändert (nicht unbedingt äquivalent zwischen 8080, 8085 und Z80)

Dezimalarithmetik verwendet wird, sondern sich auch trickreich für andere Aufgaben eignet [3], ist hier besondere Vorsicht geboten. Am sichersten ist, wenn zwischen dem Additionsbefehl, der das Hilfs-Carry-Flag verändert, und dem dazugehörigen DAA-Befehl nur Transferoperationen oder bedingte Sprünge stehen. Zu beachten ist auch, daß der DAA-Befehl nach einem Subtraktionsbefehl beim 8080 ein völlig anderes Verhalten als beim Z80 zeigt, da er für Dezimalarithmetik mit dem 8080 nur bei der Addition korrekt arbeitet. Daher darf sich der DAA-Befehl bei 8080-Programmen nur auf den ADD-, ADC- und den INC-Befehl beziehen.

Der Vollständigkeit halber sei auch auf die bei gleicher Taktfrequenz unterschiedlich lange Ausführungszeit einiger Befehle hingewiesen. *Tabelle 3* zeigt die betroffenen Befehle. Dies ist dann wichtig, wenn Zeitschleifen durch Software simuliert werden oder besonders zeitkritische Programmteile optimiert werden sollen. Interessant ist, daß einige Befehle vom 8080 schneller ausgeführt werden als von seinen Nachfolgern. Vermutlich ist auch hier als Ursache die Minimierung der Befehlscodierung zu sehen. *Marcellus Buchheit*

Tabelle 3. Befehle, deren Ausführzeiten unterschiedlich sind

Assemblerbefehle				Anzahl der Takte		
8080-Mnem.		Z80-Mnem.		8080	8085	Z80
Mo V	r,r'	LD	r,r'	5	4	4
PUSH	p	PUSH	qq	11	12	11
SPHL		LD	SP,HL	5	6	6
XTHL		EX	(SP),HL	18	16	19
PCHL		JP	(HL)	5	6	4
JNZ,JP,JNC,JC,...		JP	cc,nnnn	10/10	7/10	10/10
RNZ,RZ,RNC,RC,...		RET	cc	5/11	6/12	5/11
CALL	nnnn	CALL	nnnn	17	18	17
CNZ,CZ,CNC,CC,...		CALL	cc,nnnn	11/17	9/18	10/17
RST	n	RST	nn	11	12	11
INR	r ; DCR r	INC	r ; DEC r	5	4	4
INR	M ; DCR M	INC	(HL) ...	10	10	11
INX	q ; DCX q	INC	ss ; DEC ss	5	6	6
DAD	q	ADD	HL,ss	10	10	11
OUT	dd	OUT	(dd),A	10	10	11

r 8-Bit-Daten
q 16-Bit-Register B,D,H,SP (8080-Mnem.)
ss 16-Bit-Register BC,DE,HL,SP (Z80-Mnem.)
p 16-Bit-Register B,D,H,PSW (8080-Mnem.)
qq 16-Bit-Register BC,DE,HL,AF (Z80-Mnem.)

Literatur

[1] Z80 Assembler-Sprache. Kontron-Elektronik. Eching 1979.
[2] 8080/8085 Assembly language programming. Intel-Corporation, 1979.
[3] *Matthaei, R.-F.*: DAA – Ein Befehl mit Pfiff. Sonderheft „Mikrocomputer-Anwendungen". Franzis-Verlag, 1980.

Quadratwurzel mit dem µP Z80

Nachstehend wird ein Programm beschrieben, das mit nur 15 Bytes Code die Quadratwurzel aus einem 16-Bit-Argument errechnet.

Manche Anwendungen mit Mikrocomputern – etwa die Linearisierung von Meßwerten oder die Berechnung eines Wegs aus einer Federkraft – benötigen das Bestimmen der Quadratwurzel einer Größe. Häufig ist dabei eine 8-Bit-Auflösung des Resultats ausreichend. Es stellt sich dann die Frage nach einem geeigneten Algorithmus, der das Problem mit möglichst geringem ROM-Aufwand löst. Das Newton-Verfahren oder gar der Einbau eines Gleitkommapakets scheiden daher aus.

Algorithmus

Das hier benutzte Verfahren beruht auf der bekannten Tatsache, daß man bei fortlaufendem Addieren der ungeraden natürlichen Zahlen als Zwischensumme stets das Quadrat einer natürlichen Zahl erhält, also die Reihe 1, 4, 9, 16, 25... . Mathematisch ausgedrückt ergibt sich

$$\sum_{k=0}^{n} (2k+1) = (n+1)^2 \qquad (1)$$

Soll die Quadratwurzel aus einem Argument A bestimmt werden, sucht man die Zwischensumme, welche A am nächsten liegt. Es ist dann

$$A \approx (n+1)^2$$

und somit

$$\sqrt{A} \approx n+1$$

Die Zahl n ist im letzten verwendeten Summanden enthalten. Gemäß Gl. (1) hat dieser den Wert 2n+1.

Ein Beispiel soll den Algorithmus verdeutlichen. Es sei A = 150. Die größte Zwischensumme, die noch unter A liegt, ist 144, und der letzte Summand ist 23. Aus

$$2n+1 = 23$$

ergibt sich n+1 = 12

und damit die 8-Bit-Lösung der Wurzel aus 150.

Das Programm

Das für den Z80 geschriebene Programm 1 subtrahiert solange zunehmende ungerade Zahlen vom Argument, bis das Resultat negativ wird, das heißt es wird ein Schritt mehr als oben gezeigt ausgeführt. Der letzte Summand ist daher (2n+1)+2, und man erhält das Resultat aus

$$x = n+1 = \mathrm{INT}\left(\frac{2n+3}{2}\right)$$

Der Ausdruck auf der rechten Seite entspricht programmtechnisch einem einfachen Rechtsschieben des letzten Summanden 2n+3. Anstelle einer (beim Z80 im Gegensatz zum 8080A möglichen) 16-Bit-Subtraktion von positiven Werten werden negative Werte addiert. Dies beschleunigt die Berechnung um vier Taktzyklen pro Schleifendurchlauf.

Auf- und Abrunden

Ein Schönheitsfehler des extrem kurzen Programms sei nicht verschwiegen: Das Resultat ist stets abgerundet, was zur Folge hat, daß etwa die Quadratwurzel aus 99 als 9 bestimmt wird. Ist dies untragbar, kann man *Programm 2* verwenden, welches eine korrekte Rundung vornimmt. Dabei wird das Produkt aus dem letzten Summanden und der nächsten natürlichen Zahl gebildet und mit dem Argument verglichen. Ist das Argument größer als die Schwelle, wird das Resultat um Eins erhöht.

Rechenzeit

Die Ausführungszeit nimmt mit der Größe des Arguments zu. Bei einer Taktfrequenz von 2,5 MHz liegt sie im ungünstigsten Fall knapp über 4 ms. Eine Verbesserung ist möglich, indem man in einer Tabelle eine Anzahl von Summanden mit ihren Quadraten ablegt und mit ihrer Hilfe einen möglichst hohen Startpunkt für den Algorithmus findet.

Beide Programme sind positionsunabhängig; der Code läuft ohne Änderung überall im Speicherbereich des Z80.

Willy Kunz, Dipl. El.-Ing. ETH

```
Prog_1                                          Z-80 Assembler

Adrs  Code      Stmt    Source-Statement

                0001 ;
                0003 ;
                0004 ;
                0005 ;
                0006 ;
                0007 ;                  S Q R  1 6
                0008 ;                  ----------
                0009 ;
                0010 ; Quadratwurzel aus 16-Bit Argument ohne Rundung
                0011 ;
                0012 ;                  A := SQR(H-L)
                0013 ;
                0014 ;=============================================
                0015 ;
                0016
>0100           0017            ORG     100H
                0018
0100  11FFFF    0019 SQR16:     LD      DE,-1   ; Erste ungerade Zahl (neg.)
                0020
0103  19        0021 LOOP:      ADD     HL,DE   ; Subtrahiere ungerade Zahl
0104  3004      0022            JR      NC,ENDE ; Sprung falls fertig
0106  1B        0023            DEC     DE      ; Nächste ungerade Zahl
0107  1B        0024            DEC     DE
0108  18F9      0025            JR      LOOP    ; Wiederholen
                0026
010A  CB2A      0027 ENDE:      SRA     D       ; D-E nach A und..
010C  7B        0028            LD      A,E     ; ..durch 2..
010D  1F        0029            RRA             ; ..dividieren
010E  2F        0030            CPL             ; Entspricht NEG, DEC A
010F  C9        0031            RET             ; Das ist alles
                0032
                0033
                0034            END
```

Bild 1. Programm 1. Errechnen der Quadratwurzel mit dem µP Z80

Bild 2. ▶

Programm 2. Gegenüber Programm 1 wird hier ein korrektes Auf- bzw. Abrunden vorgenommen

```
Prog_2                                          Z-80 Assembler

Adrs  Code      Stmt    Source-Statement

                0001 ;
                0003 ;
                0004 ;
                0005 ;
                0006 ;
                0007 ;                  S Q R  1 6
                0008 ;                  ----------
                0009 ;
                0010 ; Quadratwurzel aus 16-Bit Argument mit Rundung
                0011 ;
                0012 ;  A := SQR(H-L)           H-L bleibt erhalten
                0013 ;
                0014 ;=============================================
                0015 ;
                0016
>0100           0017            ORG     100H
                0018
0100  E5        0019 SQR16:     PUSH    HL      ; Argument aufbewahren
0101  11FFFF    0020            LD      DE,-1   ; Erste ungerade Zahl (neg.)
                0021
0104  19        0022 LOOP:      ADD     HL,DE   ; Subtrahiere ungerade Zahl
0105  3004      0023            JR      NC,RUND ; Sprung falls fertig
0107  1B        0024            DEC     DE      ; Nächste ungerade Zahl
0108  1B        0025            DEC     DE
0109  18F9      0026            JR      LOOP    ; Wiederholen
                0027
010B  CB2A      0028 RUND:      SRA     D       ; D-E nach A und..
010D  7B        0029            LD      A,E     ; ..durch 2..
010E  1F        0030            RRA             ; ..dividieren
010F  2F        0031            CPL             ; Entspricht NEG, DEC A
0110  5F        0032            LD      E,A     ; Zurück nach E
0111  53        0033            INC     A       ; Nächste natürliche..
0112  14        0034            LD      L,A     ; ..Zahl in L
0113  AF        0035            XOR     A       ; Multipliziere L * E..
0114  0608      0036            LD      B,8     ; ..mit Resultat in A-L
0116  CB1A      0037 MUL8:      RR      L
0118  3001      0038            JR      NC,$+3
011A  83        0039             ADD     A,E
011B  1F        0040            RRA
011C  10F8      0041            DJNZ    MUL8
011E  CB1A      0042            RR      L
0120  67        0043            LD      H,A     ; Produkt nach H-L
0121  6A        0044            LD      A,E     ; Wurzel nach A
0122  C1        0045            POP     DE      ; Argument holen
0123  B7        0046            OR      A       ; Carry = 0
0124  ED42      0047            SBC     HL,DE   ; Vergleich mit Schwelle
0126  7B        0048            EX      DE,HL   ; Argument wieder nach H-L
0127  F0        0049            RET     P       ; Fertig falls keine Rundung
0128  3C        0050            INC     A       ; Sonst aufrunden
0129  C9        0051            RET             ; Fertig
                0052
                0053
                0054            END
```

Sonderheft:
Digitale Signalverarbeitung

Dieses Sonderheft beschreibt die Verfahren der digitalen Signalverarbeitung und die dafür benötigten Bauelemente. An Hand von Applikationen werden praktische Problemlösungen aufgezeigt. Die Fachaufsätze wurden für die Elektronik-Entwickler aus den Bereichen Telekommunikation, Unterhaltung und Industrieelektronik zusammengestellt. Da die digitale Signalverarbeitung jedoch auch Einzug in Anwendungsbereiche hält, die bisher der analogen Signalverarbeitung vorbehalten waren, ist dieses Sonderheft ebenfalls für klassische Analog-Entwickler interessant.

BASIC-Programm berechnet Fehler eines D/A-Umsetzers

Der folgende Beitrag beschreibt eine nützliche Anwendung für Basic-Rechner im Taschenformat.

Der Gesamtfehler eines D/A-Umsetzers wird im wesentlichen durch die Summe der drei Komponenten Offset-Fehler, Verstärkungsfehler und Nichtlinearitätsfehler bestimmt. Im Gegensatz zu den ersten beiden läßt sich der Nichtlinearitätsfehler nicht abgleichen. *Bild 1* zeigt die Meßschaltung zur Bestimmung des Nichtlinearitätsfehlers eines 12-Bit-Komplementär-Binär-Umsetzer (z. B. DAC 87, DAC-HY 12 BC, DAC 80). Durch Anlegen von „0"-Potential am Eingang eines beliebigen Bits wird der entsprechende Analogschalter im Umsetzer durchgeschaltet. Die Ausgangsspannung wird mit Hilfe eines Digitalvoltmeters gemessen. Eine Besonderheit bei der Berechnung des Nichtlinearitätsfehlers (*Bild 2*) ist die automatische Korrektur der Einzelmeßwerte auf die optimale Übertragungsfunktion durch das Programm, so daß ein Abgleich des Offsets und der Verstärkung vor der Messung nicht notwendig ist.

Die erforderliche Genauigkeit des DVM wird durch die Auflösung des Umsetzers bestimmt. Das Auflösungsvermögen eines 12-Bit-D/A-Umsetzers für 1/2 LSB beträgt 0,012 % vom vollen Ausgangsspannungsbereich. Die Genauigkeit des DVM sollte um den Faktor 10 größer sein. Deshalb ist der Einsatz eines 5stelligen Geräts mit einem Fehler von 0,001 % für die Spannungsmessung sinnvoll. Der Umsetzer muß mit einer stabilen Gleichspannungsquelle (±0,1 %) versorgt werden. Besondere Aufmerksamkeit ist auf die Verlegung der Masseleitung zu richten.

Für die Fehlerberechnung gelten folgende Formeln:

$$D = \frac{U_{FS(real)} - U_{FS(ideal)}}{U_{FSR}} \cdot 100 \qquad (1)$$

$$G = \frac{\left[\sum_{i=1}^{n} U_{i(real)} - n \cdot U_{FS(real)}\right] - \left[U_{FSR} - \frac{U_{FSR}}{2^n}\right]}{U_{FSR}} \cdot 100 \qquad (2)$$

$$F_i = \frac{1}{1 + \frac{G}{100}} (U_i - U_{FS(real)}) - \frac{U_{FSR}}{2^i} \qquad (3)$$

$$E_i = \frac{F_i \cdot 2^n}{U_{FSR}} \qquad (4)$$

$$H = \sum_{i=1}^{n} E_i \, (E_i > 0) \qquad (5)$$

Dabei bedeuten im einzelnen:

U_{FSR} = Bereich der vollen Ausgangsspannung
$U_{FS\,(ideal)}$ = ideale Ausgangsspannung
$U_{FS\,(real)}$ = reale Ausgangsspannung, d. h. Offset und Verstärkung nicht abgeglichen
D = Offset-Fehler in % von U_{FSR}
G = Verstärkungsfehler in %
F_i = Bitfehler von Bit_i in V
E_i = Bitfehler von Bit_i, bezogen auf LSB
H = Totaler Linearitätsfehler, bezogen auf LSB
U_i = Ausgangsspannung bei Bit_i
n = Anzahl der Bits

Das Programm ist in BASIC auf einem Taschencomputer des Typs PC 1210 (Sharp) geschrieben und bis auf ein paar Sonderbefehle auf den meisten BASIC-Systemen lauffähig. Nach der Eingabe der Werte U_{FSR}, $U_{FS\,(ideal)}$, $U_{FS\,(real)}$ und der Anzahl der Bits

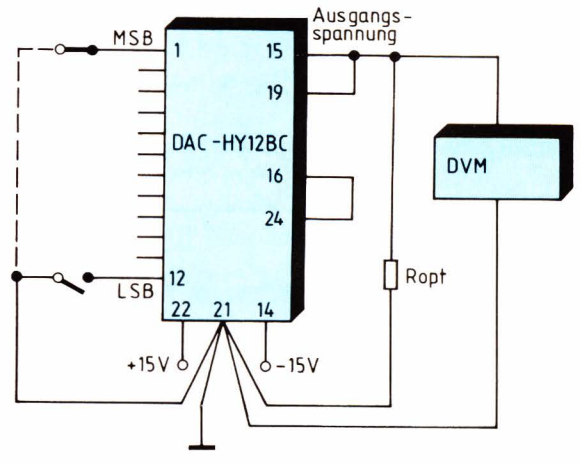


Bild 1. Meßaufbau zur Fehlerbestimmung eines D/A-Umsetzers: Beginnend mit MSB werden nacheinander alle Schalter geschlossen

```
10 'A':INPUT "FSC= ";A,"IDEAL ";B,"REAL ";C,"?-BIT ";N
20 I=0:B=ABS (B):C=ABS (C):M=14+N:J$="FEHL."
30 FOR L=15TO M:GOSUB 90:INPUT A(L):I=I+A(L):NEXT L
40 D=(B-C)/A*100:PRINT D;"(%)OFFS-";J$:I=ABS (I)
50 G=(N*C-I)-(A-A/2∧N)/A*100:H=ABS (G):PRINT H;"(%)VERST.";J$:H=0
60 FOR L=15TO N:K=ABS (A(L)):F=(C-K)/(1+G/100)-(A/2∧(L-14)):GOSUB 90:PRINT F;" (V)BIT-";J$
70 E=F*2∧N/A:PRINT E;" (LSB)-";J$:D=SGN (E):IF DLET H=H+E,
80 NEXT L:PRINT H;" TOT.LIN-";J$:END
90 USING :PAUSE (L-14):BEEP 1:USING "##.###":RETURN
```

Bild 2. Dieses BASIC-Programm berechnet Fehler von D/A-Umsetzern

werden die Einzelmeßwerte vom Rechner abgefragt. Nach der Eingabe des letzten Wertes werden der Offset- und der Verstärkungsfehler berechnet. Danach können Einzelfehler der Bits bezogen auf die Ausgangsspannung und auf das LSB abgefragt werden. Als letzter Wert wird der Nichtlinearitätsfehler berechnet. Eine Besonderheit der Programmsprache des PC 1210 ist in Zeile 10 das in Hochkommas eingeschlossene A, durch das im DEF-Betrieb der Aufruf des Programmes durch Shift A erfolgt. Das Zeichen „∧" in diversen Programmzeilen heißt potenzieren. USING ist eine Formatanweisung, und in Zeile 70 ('IF D LET...') findet sich ein Kürzel für IF D > 0.

Ing. (grad.) Horst Zander

Literatur

[1] *Jacobs, R. W.:* Specifying & Testing Digital to Analog Converters. Appl. Bulletin AN-25, Fa. Teledyne Philbrick, July 1976.

[2] *Prazak, P.:* Programmable Handheld Calculator computes Digital-To-Analog Converter Errors. Application Note AN-94, Fa. Burr-Brown, November 1978.

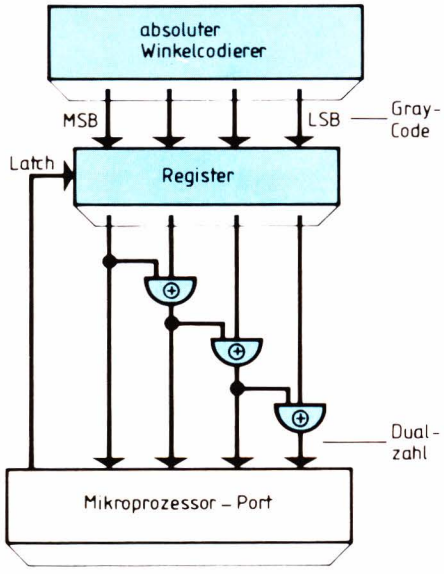
Programm zur Gray-Code-Decodierung

Die Hardware zur Gray-Code-/Dualzahl-Umwandlung kann ersetzt werden durch Software. Das hier beschriebene Kleinstprogramm dient diesem Zweck und ist für Mikrorechner besser geeignet als eine direkte softwaremäßige Nachbildung der Hardware.

Für die digitale Messung von Drehwinkelpositionen werden neben den inkrementalen auch häufig absolute Winkelcodierer eingesetzt, die einen zum Drehwinkel proportionalen Zahlenwert im Gray-Code [1] abgeben. Der Gray-Code hat gegenüber dem Binärcode den Vorteil, daß sich zwei aufeinanderfolgende Zahlenwerte jeweils nur in einer Stelle unterscheiden; dadurch können beim kontinuierlichen Verändern des Winkels keine falschen Zwischenwerte auftreten.

Zahlen im Gray-Code sind für Rechenoperationen nicht geeignet. Man benötigt daher eine Gray-Code-/Dualzahl-Umwandlung. Die übliche Hardware-Realisierung [1] zeigt *Bild 1* am Beispiel der Wortlänge n=4. In der Praxis findet man Werte von n=6...14. Will man die umgewandelte Dualzahl in einen Rechner einlesen, so muß der eigentlichen Decodierlogik aus Exklusiv-ODER-Gattern ein Register gemäß Bild 1 vorgeschaltet werden, um falsche Zwischenwerte aufgrund der XOR-Gatter-Laufzeiten zu vermeiden.

Will man nun die Hardware aus Bild 1 durch Software ersetzen, so führt dies zunächst auf eine mit Mikrorechnern recht mühsame bitweise Verarbeitung. Das nachstehend beschriebene, von einem bestimmten Prozessortyp unabhängige Kleinstprogramm basiert dagegen auf einer wortweisen Verar-



ⓐ

```
                LD  X  :   X ──► A   (Register A laden)
Schleifen-      LRS    :   A/2 ► A   (Logisches Rechtsschieben)
durchlauf       XOR X  :   A⊕X► A    (Exklusiv-ODER)
m-mal           ST  X  :   A ──► X   (Abspeichern nach X)
```

ⓑ

n	2	3	4	5	6	7	8	9	10	11	...
m	1	3	3	7	7	7	7	15	15	15	...

Bild 2. Umwandlungsprogramm: a) Programmschleife, b) Anzahl der notwendigen Schleifendurchläufe m in Abhängigkeit von der Wortlänge n der umzuwandelnden Zahl

◀ Bild 1. Hardware-Decodierer

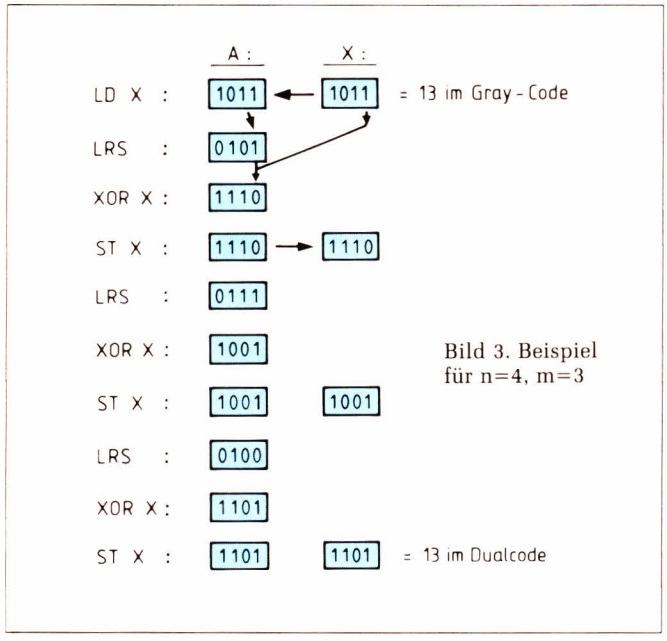


Bild 3. Beispiel für n=4, m=3

beitung (*Bild 2*). Es arbeitet also ganz anders als die Hardware, leistet jedoch dasselbe; die Theorie dazu findet sich in [2].

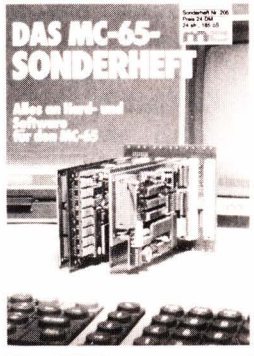
Die Speicherzelle X enthalte rechtsbündig die umzuwandelnde Zahl im Gray-Code; A sei ein Arbeitsregister. *Bild 2a* zeigt die sehr einfache Programmschleife, die die Umwandlung ausführt. Die Anzahl m der notwendigen Schleifendurchläufe ist um eins kleiner als die kleinste Zweierpotenz, die größer gleich der Wortlänge n ist (*Bild 2b*). Ein Zahlenbeispiel für n=4 ist in *Bild 3* angegeben. Im übrigen und nur nebenbei sei angemerkt, daß ein einmaliger Schleifendurchlauf für alle Wortlängen n die umgekehrte Code-Umwandlung, also vom Dual- zum Gray-Code, ermöglicht.

Dr.-Ing. Herbert Wehlan

Literatur

[1] *Tietze, U., Schenk, Ch.*: Halbleiter-Schaltungstechnik. 4. Auflage, Springer-Verlag, Berlin 1978, S. 465.

[2] *Wehlan, H.*: Gray-Code-/Binärzahl-Umwandlung per Software. Regelungstechnik 28 (1980), Heft 2, S. 61...63.



Das MC-65-Sonderheft

Das MC-65-System wurde 1984 in mc vorgestellt. Dieses Sonderheft nun enthält die gesamte Hard- und Software, die zum Aufbau des MC-65 erforderlich ist. Die meisten Beiträge wurden in mc bereits veröffentlicht, sind aber hier auf den neuesten Stand gebracht und überarbeitet. Neu hinzugekommen ist beispielsweise ein komfortabler Bildschirmeditor, der die Anwendungsmöglichkeiten des MC-65 ganz wesentlich erweitert. Bezugsmöglichkeiten: bei allen Bahnhofsbuchhandlungen, beim Elektronik-Fachhandel, bei größeren Zeitschriftenverkaufsstellen, in Buchhandlungen oder direkt beim Franzis-Verlag gegen Zusendung eines Schecks (DM 26,–).

Wenn Sie die Kommunikations- und Unterhaltungselektronik immer im Auge behalten wollen:

Die Funkschau informiert in Fachbeiträgen alle 14 Tage umfassend zu den Themen Kommunikations- und Computertechnik, Audio und Video.

Technische Beiträge

Beschreibung von Geräten, Systemen und ihren Funktionen – einschließlich technischer Details, wenn sie für den Anwender von Bedeutung sind.

Erfahrungsberichte

Geräte der elektronischen Kommunikationstechnik und der Konsumelektronik im praktischen Gebrauch. Die FUNKSCHAU beschreibt den Nutzen für den Anwender.

Produktneuheiten

Das Schaufenster des Elektronik-Marktes. Neue Produkte werden mit den wichtigsten technischen Daten, Anwendungsbeispielen und Bezugsquellenangaben präsentiert.

Service-Beiträge

Arbeitshilfe für Service-Techniker aller Elektronikbereiche: Arbeitsblätter mit Grundschaltungen der Elektronik, Schaltungsdetails, Schaltungsapplikationen, Meßtechnik, Fehlerquellensuche...

Bauanleitungen

Für den anspruchsvollen Hobby-Elektroniker zum sicheren Nachbau. Beispiele: Meßgeräte, Videomischpulte, Heizungsregelung per Computer, Alarmanlagen, Musiksynthesizer...

Reports

Aktuelle Elektronik-Themen aus unterschiedlichen Blickwinkeln betrachtet. Runduminformationen, die neben der Technik auch gesellschaftspolitische Aspekte neuer Entwicklungen beleuchten.

Nachrichten

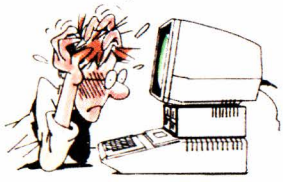
Was für den Leser zur Beurteilung der Technik des Elektronik-Marktes wichtig ist, wird übersichtlich aufbereitet und mit Stellungnahmen ergänzt.

Wenn Sie sich vom Nutzen der FUNKSCHAU für Ihre Arbeit überzeugen wollen, schicken wir Ihnen gerne im Rahmen unseres Kennenlern-Angebots die neueste Ausgabe kostenlos und unverbindlich ins Haus.
Die vorbereitete Karte finden Sie an der Umschlagklappe.

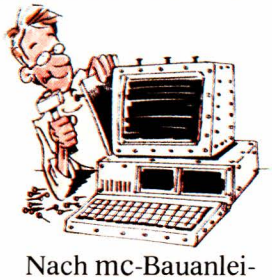
Funkschau
Zeitschrift für Unterhaltungselektronik und Kommunikationstechnik

Für engagierte Anwender:

Mit mc lernen Sie Computer von Grund auf verstehen. Ausführliche Funktionsbeschreibungen von Rechner-Hardware und gut kommentierte Programm-Listings bieten Ihnen den richtigen Einstieg ins ernsthafte Computern.



Durch Programme in mc werden Sie manches Problem überhaupt nicht mehr als Problem betrachten.



Nach mc-Bauanleitungen löten Sie vom einfachen Interface bis zum kompletten System, was an Hardware nur schwer zu kaufen ist.

In mc-Fachaufsätzen geht's um neue Entwicklungen, um professionelle Hardware und Peripherie.

Natürlich testet mc Geräte und Programme. Die Ergebnisse werden aus der Sicht des professionellen Anwenders interpretiert.

Aktuelles aus der Branche zu Unternehmen, Produkten, Kongressen, Tagungen und Messen finden Sie jeden Monat in mc.

mc bringt Profis weiter.
Für DM 6,50 bekommen Sie mc an jeder größeren Zeitschriften-Verkaufsstelle.
mc können Sie aber auch auf andere Art kennenlernen.
Kostenlos und unverbindlich.
Die Abrufkarte dafür finden Sie an der Umschlagklappe.

mc
Die Mikrocomputer-Zeitschrift